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All rights reserved as an unpublished work.

This is an unapproved and unpublished IEEE Standards Draft, subject to change.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

4.6 Transaction Processing Services

Responsibility: Bob Gambrel

4.6.1 Overview and Rationale

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

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

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

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

4.6.2 Scope

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

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

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

2075 — Transaction access to nondatabase ''resource managers'' (such as the
2076 software managing the cash in an automatic teller machine)

2077 — Distributed Databases—databases spanning multiple machines, but
2078 accessed by application programs as if just a single database

2079 — Online Transaction Processing (OLTP)—the scheduling of ''transaction pro-
2080 grams'' based on terminal input with consolidated recovery of the database
2081 updates and the terminal messages

2082 — Distributed Transaction Processing (DTP)—different machines running
2083 multiple application programs with multiple databases, using a
2084 client/server or conversational application-to-application communications
2085 paradigm

2086 Note that Transaction Processing Services are used in all of the above situations,
2087 and others too.

2088 Finally, it should be noted that ''transactions'' are not really ''messages,'' but
2089 rather ''units of work'' that may encompass multiple messages. Furthermore,
2090 while traditionally ''transaction processing'' has usually been synonymous with
2091 ''OLTP'' where so-called ''immediate transactions'' are the norm, other types of
2092 transactions are also covered: ''batch transactions'' (where the work is done in the
2093 ''background'') and ''deferred transactions'' where there may be a time dependence
2094 on the transaction, such as a fixed start time.

2095 This clause addresses the current work in progress in groups such as ISO and oth-
2096 ers.

2097 ### 4.6.3 Reference Model

2098 This subclause addresses the conventional Transaction Processing Reference
2099 Model, the POSIX OSE Reference Model (incorporating transaction processing con-
2100 siderations), and other important real world considerations introduced by Tran-
2101 saction Processing.

2102 #### 4.6.3.1 Conventional Transaction Processing Reference Model

2103 A model for transaction processing is developed here to complement the POSIX
2104 system model. Current work in progress by the POSIX. 11 Transaction Processing
2105 Working Group and other groups such as ISO/IEC JTC 1/SC21 Open Systems
2106 Interconnection—Distributed Transaction Processing may result in a Transaction
2107 Processing Reference Model more suitable than the one developed here. At that
2108 time, such a model will be referenced and incorporated into the POSIX OSE refer-
2109 ence model. Until that time, the current model will be used as a convenient
2110 means for describing the services needed in this domain.

2111 While transaction processing services have usually been thought of as applying to
2112 databases, the applicability goes further. Nonetheless, the description given here
2113 of the transaction processing model shows how the transaction processing pro-
2114 gram can view the transaction services as an extension of the the Database view

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

of the POSIX OSE reference model as shown in Figure 4-10. From the transaction application program point of view, a transaction processing system has additional capabilities that go beyond those provided by database systems. These services to the transaction application program are provided at an API that is called the *Transaction Manager API.* (See Figure 4-13.) For convenience in discussing the model, the provider of those services is called the *Transaction Manager* (TM).

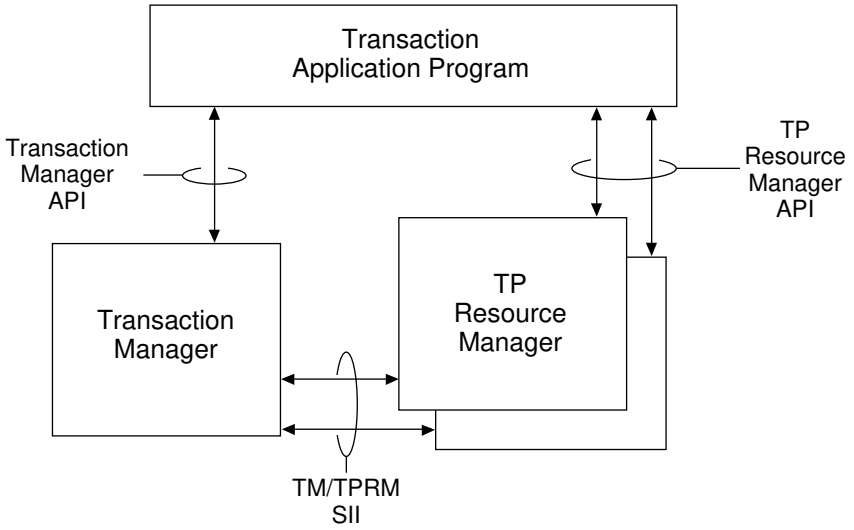

Figure 4-13 – The Conventional Transaction Processing Model

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

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

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

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

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

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

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

4.6.3.2 POSIX OSE Reference Model (with Transaction Processing)

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

4.6.3.3 Implementation Aspects E

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

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

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

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

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

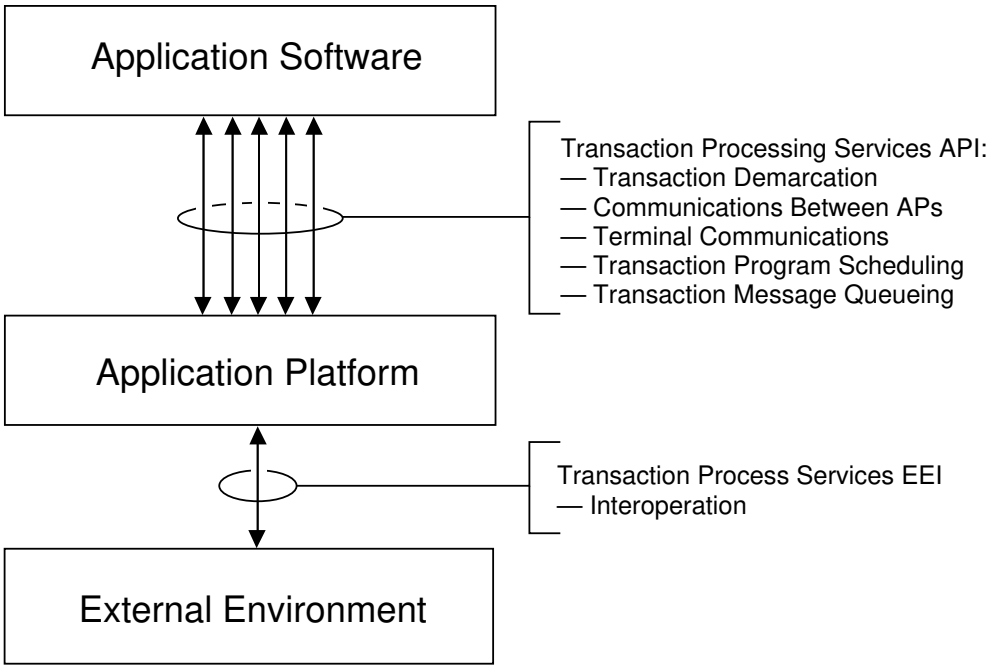

Figure 4-14 – POSIX OSE Transaction Processing Reference Model

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

4.6.4 Service Requirements

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

4.6.4.1 Application Program Interface Services

E

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

Transaction Demarcation

— Indicate the start of a transaction.

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

— Indicate a transaction has ended successfully (commit) or unsuccessfully (rollback).

— Indicate the beginning and ending of nested ''subtransactions'' whose commitment is independent of the ''parent transaction''. (Nested within a parent transaction can be multiple subtransactions. Subtransactions are independent of each other, and whether subtransactions commit or not does not affect the commitment of the parent.)

— Suspend and resume transaction mode (to do work which is not be committed or rolled back when the transaction is completed). This can be thought of as nesting nontransaction work within a transaction.

E

Communications Between Transaction Application Programs

— Call another transaction application program (possibly remote) within the context of a transaction.

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

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

Terminal Communications

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

Transaction Program Scheduling

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

Transaction Message Queuing

— Define a ''message'' (based, possibly, on screen input from the end user) that, from the application point of view, precisely defines a unit of work to be done by this transaction or another transaction.

— ''Send'' a message to another transaction application program.

— Retrieve the next message (and then act upon it)

— Prioritize and associate start times with messages

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

NOTE: Several of the above services are similar to but semantically different from similar sounding

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

services in other clauses of this section. They are listed here because they are ''transactional''; i.e., the concept of a transaction and the ACID properties are provided by these services.

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

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

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

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

— Determine the amount of a valuable resource that remains. E

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

4.6.4.2 External Environment Interface Services E

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

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

E

4.6.4.3 OLTP Resource Management Services

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

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

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

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

— Modeling Services — Predict the system resources needed to process a given transaction processing workload.

— Directory/Namespace Services — Map names to addresses.

— Recovery/Restart Services — Recover and restart transactions involving one or more transaction application programs using one or more TP Resource Managers.

— Test Services — Automatically generate tests for workload simulation, etc.

— System Configuration Services — Replace or add transaction application programs without the need to shut down the execution environment.

E

4.6.5 Standards, Specifications, and Gaps

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

Table 4-7 – Transaction Processing Standards

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

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

4.6.5.1 Current Standards

None. E

4.6.5.2 Emerging Standards

OSI Distributed Transaction Processing (DTP)

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

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

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

Table 4-8 – Transaction Processing Standards Language Bindings

Standard	LIS	Ada	APL	BASIC	C	C++
POSIX. 11	E				E	

Standard	COBOL	C-LISP	Fortran	Pascal	PL/1	Prolog
POSIX. 11						

NOTES: LIS — Language-independent specification is available.

Ada, APL, BASIC, — Language-dependent specifications exist.

S, E, G — Standard, Emerging Standard, Gap

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

POSIX. 11 POSIX Transaction Processing

POSIX. 11

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

— Transaction Manager (TM) Services provided at the Transaction API.

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

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

4.6.5.3 Gaps in Available Standards

4.6.5.3.1 Public Specifications

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

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

X/Open TP

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

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

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

4.6.5.3.2 Unsatisfied Service Requirements

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

4.6.6 POSIX OSE Cross-Category Services

Not applicable.

4.6.7 Related Standards

CCR

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

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

See 4.3 for more information.

E

SQL Standard Database Language

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

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

See 4.4

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

4.7 Graphical Window System Services

Responsibility: Marti Szczur and Ruth Klein

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

4.7.1 Overview and Rationale

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

— The existence and popularity of windowing systems

— The requirement for development of applications that take advantage of the windowing system environment

— The requirement of many users and manufacturers for a basic consistency in the presentation and behavior of graphical window systems across multiple graphics platforms

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

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

4.7.2 Scope

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

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

— Application Program Interfaces (API) for such protocols

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

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

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

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

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

Standards relating to the following are not considered:

— Graphics; see 4.8.

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

— Network transport protocols; see 4.3.

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

4.7.3 Reference Model

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

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

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

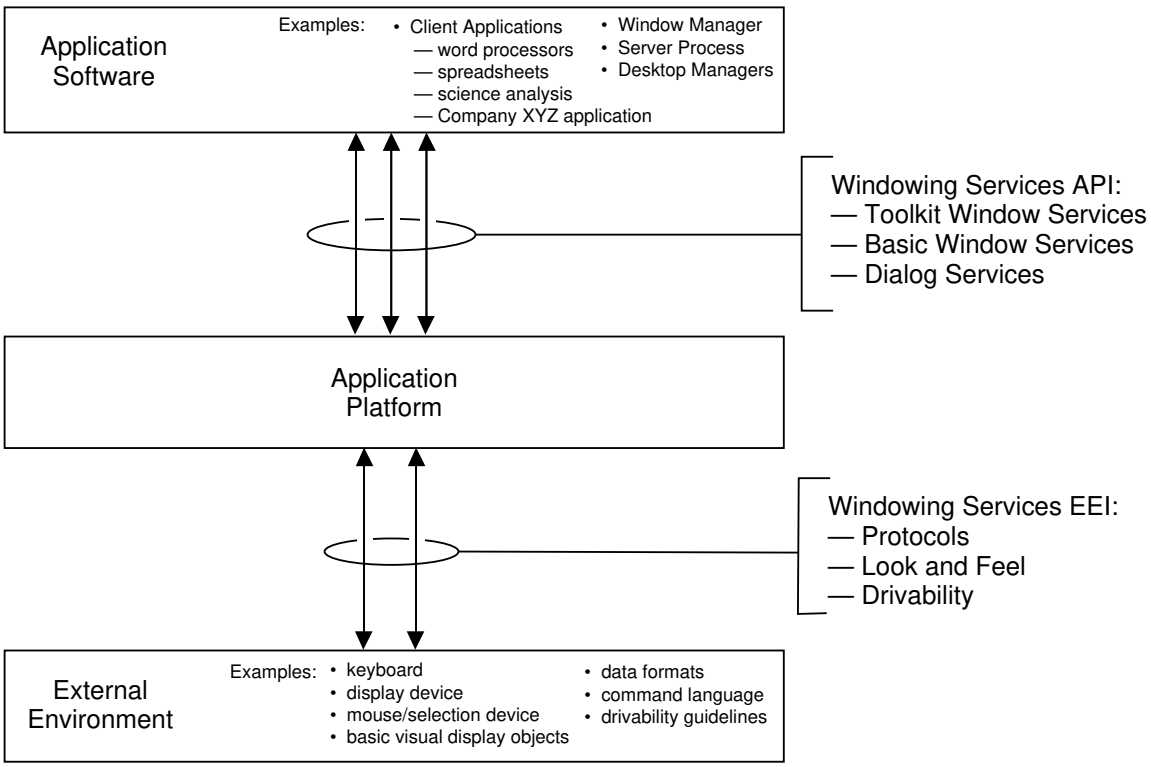

Figure 4-15 – Windowing Reference Model

which will eliminate problem areas such as:

— Error provoking inconsistencies

— Misleading expectations about the results of user actions

— Gross inconsistencies in the high level user model or metaphor

— Incompatible motor control tendencies

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

The API is concerned with the interface between the application semantics and the graphical window system services. It is the interface between the application

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

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

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

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

4.7.3.1 Application Software Elements

Application Entity Elements include:

(1) Window System Server

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

(2) Window Manager

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

— Resize window

— Move window

— Push/pop window to top/bottom

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

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

(3) Local and Remote Applications

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

4.7.3.2 Application Program Interface (API) Elements

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

4.7.3.3 External Environment Interface (EEI) Elements

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

External Environment Interface Elements include:

— Display Device Specifications

— Data Protocol Format

— User Drivability Guidelines (e.g., ''look and feel'' of window interface)

— Keyboard Device Specification

— Selection Device Specification (e.g., mouse, graphics tablet, touch screen)

— Command-language Definition (syntax and semantics guidelines)

4.7.4 Service Requirements

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

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

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

4.7.4.1 Application Program Interface Services

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

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

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

4.7.4.1.1 Basic Window Services

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

The major functional areas are:

— Window Management

— Presentation Management

— Event Handling

— Error Handling

— Interclient Communications

— Input Device Management: Keyboard, Pointing Device

— Screen Management

— User Preferences Management

— Server Connection Management

The following functions are available under each functional area.

Window Management

Functions available for Window Management are:

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

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

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

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

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

Presentation Management

Functions available for Presentation Management are:

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

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

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

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

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

— Draw text and obtain text metric information

Event Handling

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

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

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

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

Functions available for Event Handling are:

— Select events

— Map events to a window

— Get information about events

— Send events

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

Error Handling

Functions available for Error Handling are:

— Get error message

— Get error description

— Set error event handler routine

Interclient Communication

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

— Properties

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

— Selections

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

— Cut Buffers

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

Functions available for Interclient Communication are:

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

— Set and get selections

— Manipulate cut buffers

Input Device Management

Functions available for Input Device Management are:

— Receive keyboard input and pointing device button events

— Gain exclusive control of keyboard or pointing cursor

— Track the pointing cursor

— Change Server-wide keyboard mappings

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

— Set and get keyboard and pointing device preferences

Screen Management

Functions available for Screen Management are:

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

— Get, display, and manipulate bitmapped screen images

— Screen saver functions (blanking screen on idle)

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

User Preferences Management

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

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

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

— The application program's defaults

— The command line arguments

Functions available for User Preferences Management are:

— Set and get preference data

Server Connection Management

Functions available for Server Connection Management are:

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

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

— Obtain Server implementation information

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

4.7.4.1.2 Toolkit Window Services

The Toolkit Window services provide a mechanism for runtime access to a library of visual objects. A visual object is a graphical display object (i.e., interaction object) with associated software that receives input from users (typically via a keyboard and a pointing device) and communicates with applications and other visual object software. The graphical representation of a visual object can be

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

modified to reflect the results of application processing. Examples of visual objects are graphical push buttons, check boxes, and editing boxes. (Note: The term used within the X Window System community to define visual objects is ‘‘widgets.’’)

Toolkit Window services are provided for two reasons:

— To allow application software to directly utilize a visual object library

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

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

An application may use Toolkit Window services to:

— Perform toolkit initialization/exit

— Set up visual object resources

— Create/delete a visual object

— Display a visual object

— Add/remove application-specific routines to be called by a visual object (event callbacks)

— Retrieve/modify the state of a visual object

— Turn control over to the toolkit for user input processing

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

— Manage child visual objects (a child visual object is a visual object that is displayed inside of another visual object)

— Manage window events, timer events, and file input events

— Handle visual object geometry (sizing, positioning, child visual object placement)

— Handle user input

— Manage visual object resources

— Translate an event into an action

— Manipulate graphics contexts

— Manipulate pixmaps (pixel arrays—used to display a graphical object by turning pixels on and off)

— Manage memory associated with graphical window systems

— Handle errors associated with graphical window systems

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