-- --

# IEEE P1003.0 Draft 13 – September 1991

## 4.5 Data Interchange Services

*Responsibility: Richard Scott*

### 4.5.1 Overview and Rationale

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

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

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

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

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

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

### 4.5.2 Scope

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

### 4.5.3 Reference Model

The Data Interchange Services relate directly to the POSIX Open System Environment reference model that was presented in Figure 3-1. Figure 4-13 shows the components of the reference model that are significant for data interchange. The reference model defines the conceptual relationships required to provide these facilities. It should not be viewed as a description of an implementation. The key entities within the figure are the Application Software, the Application Platform, and the External Environment. To satisfy the data interchange service requirements, the POSIX Open System Environment must permit application software to transfer data to and from the external environment.

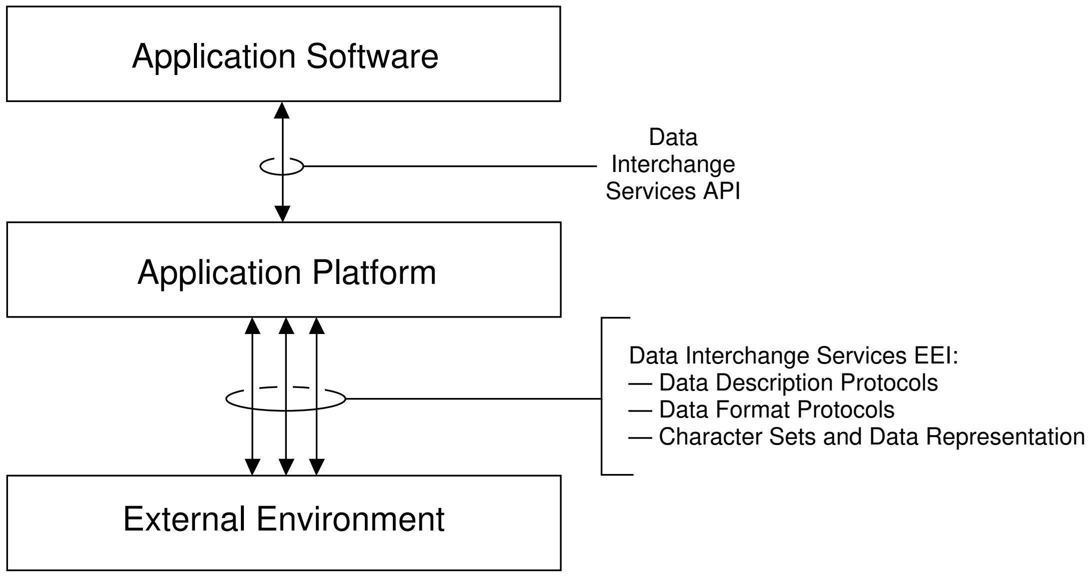


**Figure 4-13 – Data Interchange Reference Model**

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

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

### 4.5.4 Service Requirements

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

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

#### 4.5.4.1 Application Program Interface Services

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

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

#### 4.5.4.2 External Environment Interface Services

This section identifies the EEI services required to support data interchange. These services are all in the form of protocol and format definitions. As shown in D
Figure 4-13, these protocols include:

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

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

##### 4.5.4.2.1 Character Sets and Data Representation

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

##### 4.5.4.2.2 Data Format Protocols

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

— Detailed storage format for the value

— The data value in an environment-neutral format

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

#### 4.5.4.2.3 Data Description Protocols

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

## 4.5.5 Standards, Specifications, and Gaps

See Table 4-8.

### 4.5.5.1 Current Standards in the POSIX OSE

#### 4.5.5.1.1 International Standards

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

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

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

**Standard Generalized Markup Language (SGML)**

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

**Table 4-8 – Data Interchange Standards**

| Service | | Specification | Subclause | |
|---|---|---|---|---|
| Data Description Protocols | ODA/ODIF | ISO 8613 Parts 1-8 | 4.5.5.? | D |
| | SGML | ISO 8879 | 4.5.5.? | D |
| | EDIFACT | ISO 9735 | 4.5.5.? | D |
| | STEP | ISO DP 10303 | 4.5.5.? | D |
| | EDIFACT | ANSI X.12 | 4.5.5.? | D |
| | IGES | NBSIR 86 | 4.5.5.? | D |
| | VHDL VHSIC | IEEE P1076 | 4.5.5.? | D |
| Data Format Protocols | ODA/ODIF | ISO 8613 Parts 1-8 | 4.5.5.? | D |
| | SGML | ISO 8879 | 4.5.5.? | D |
| | CGM | ISO 8632 | 4.5.5.? | D |
| | CGM | ANSI X3.122-1986 | 4.5.5.? | D |
| | STEP | ISO DP 10303 | 4.5.5.? | D |
| | EDIFACT | ISO 9735 | 4.5.5.? | D |
| | EDIFACT | ANSI X.12 | 4.5.5.? | D |
| | IGES | NBSIR 86-3359 | 4.5.5.? | D |
| | VHDL VHSIC | IEEE P1076 | 4.5.5.? | D |
| | CDIF | | 4.5.5.? | D |
| | GDSII | | 4.5.5.? | D |

**Computer Graphics Metafile (CGM)**

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

**Electronic Data Interchange (EDI)**

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

#### 4.5.5.1.2 Regional Standards

None.

#### 4.5.5.1.3 National Standards

None.

### 4.5.5.2 Emerging Standards in the POSIX OSE

#### 4.5.5.2.1 International Standards

**Standard for the Exchange of Product Model Data (STEP)**

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

#### 4.5.5.2.2 Regional Standards

None.

#### 4.5.5.2.3 National Standards

None.

### 4.5.5.3 Gaps in Available Standards

#### 4.5.5.3.1 Standards and Specifications Outside the POSIX OSE

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

##### 4.5.5.3.1.1 Government/Legal Standards

**Initial Graphics Exchange Specification (NBSIR 86-3359)**

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

##### 4.5.5.3.1.3 De Facto Standards

**CASE Data Interchange Format (CDIF)**

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

**Graphic Design System II (GDSII) CAD Exchange Format**

CAD exchange format.

**Hardware Description Language (VHDL VHSIC)**

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

#### 4.5.5.3.2 Unsatisfied Service Requirements

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

### 4.5.6 OSE Cross-Category Services

Not applicable. D

### 4.5.7 Related Standards

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

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

### 4.5.8 Open Issues

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

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

## 4.6 Windowing System Services D

*Responsibility: Marti Szczur and Ruth Klein*

### 4.6.1 Overview and Rationale

Human user interface standards are considered in this clause. The user interface is a key component of all computer systems that support direct human-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 lead to the proliferation of graphical user interfaces (GUI s), windowing technologies, generic commands, and an assortment of selection techniques (e.g., mouse, track ball, tablets). In several of these technologies de facto standards are emerging and becoming informally accepted by the user community, and with more frequency, mandated for use in systems being developed within government agencies and private industry. The primary motivations for considering human/computer interaction 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 human computer interaction 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). HCI services certainly utilize graphic technologies, but are limited to providing graphics related to window-based user interfaces and specifications on how human users may interact with an application within a window environment. The graphic services are addressed independently in 4.7.

Performance metrics are used in the specification of HCI components. It is intended that (eventually) a set of metrics that will be sufficient to characterize all of the services defined above.

— Screen width—effective horizontal width of screen

— Screen height—effective vertical height of screen

— Screen depth—effective depth of screen (i.e., bit width, gray scale)

— Screen resolution—size in units of $10^6$

— Application display latency—time to satisfy a display request measured in units of milliseconds; the interval begins with release of the request from the (possibly remote) application. The interval ends with completion of requested service.

### 4.6.2 Scope

Standards and standards initiatives in the user 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

— HCI drivability features that define a common subset of ''look and feel''; i.e., appearance, screen positioning, and behavior of human/computer interaction 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 interface that may be entered interactively by the human 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 4.7.

— Keyboard layout (out of scope for HCI Services)

— Network transport protocols; see 4.3.

— Hardware device interfaces (out of scope for HCI Services)

### 4.6.3 Reference Model

This subclause identifies the entities and interfaces specific to the construction of an HCI architecture. This architecture is consistent with, and extends the architecture of, Section 3. As illustrated in Figure 4-14, 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).

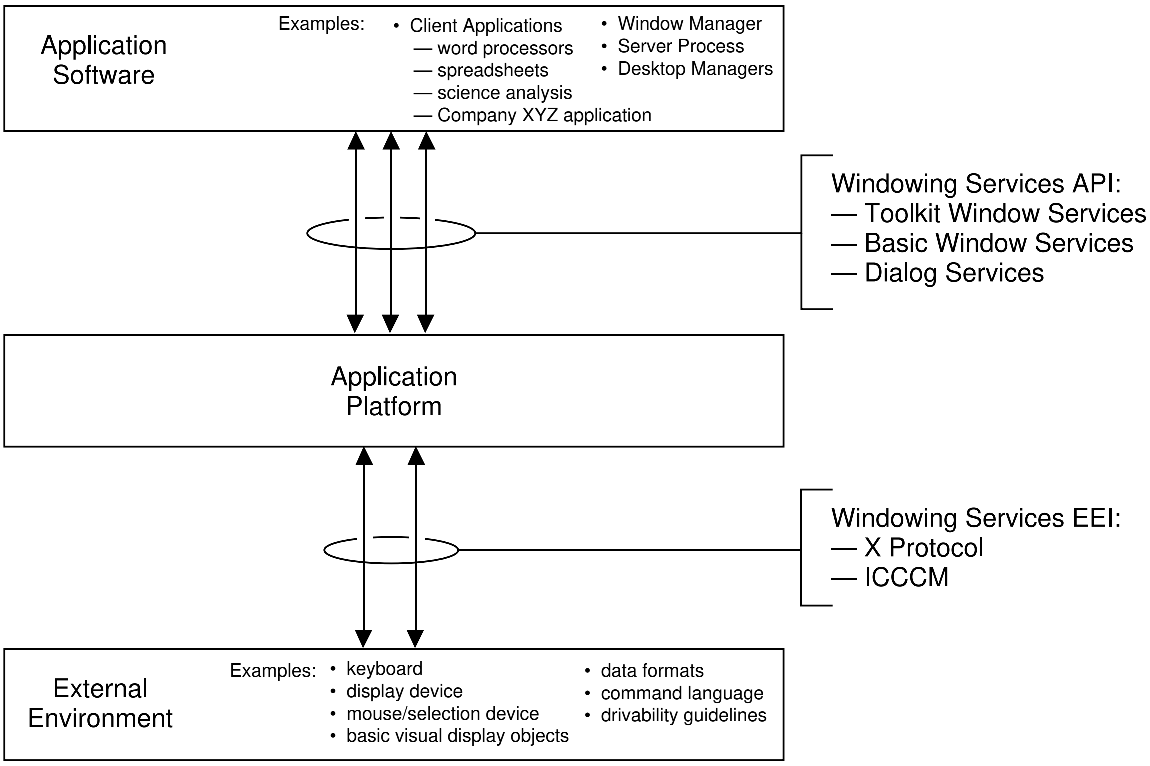


**Figure 4-14 – Windowing Reference Model** D

The EEI is concerned with the communication with the user via the physical HCI 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 human user interfaces, 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 human user interfaces 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 HCI services. It is the interface between the application software and the application platform and is defined primarily in support of application portability. These services provide functions for creation and manipulation of visual display objects such as menus, buttons, scrollbars, and dialog boxes. In addition, these functions allow information about user actions to flow back to the application software; for example, when the user has selected an item from a menu. This information about user actions is known as an event. Applications that require communication with the human 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 human users to easily operate the application across platforms.

Elements of the HCI 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 HCI entities, interfaces, services, and standards.

#### 4.6.3.1 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 human user capabilities shall include but are not limited to:

— Resize window

— Move window

— Push/pop window to top/bottom

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

(3) Local and Remote Applications

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

#### 4.6.3.2 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.

#### 4.6.3.3 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)

### 4.6.4 Service Requirements

Human-Computer Interaction (HCI) Services provide a controlled interface between the application-specific software and the user-interface-specific (i.e., HCI) 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 human interface communication. A common, standardized human/computer interaction for applications shall be available to users across all POSIX Open Systems Environments.

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

### 4.6.4.1 Application Program Interface Services

Application services include those services made available to the application developer to separate the application functions from the HCI 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 shall have access to the following services from an application via language bindings.

#### 4.6.4.1.1 Basic Window Services

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

The major functional areas are:

— Window Management

— Presentation Management

— Event Handling

— Error Handling

— Interclient Communications

— Input Device Management: Keyboard, Pointing Device

— Screen Management

— User Preferences Management

— Server Connection Management

The following functions are available under each functional area.

**Window Management**

Functions available for Window Management are:

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

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

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

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

**Presentation Management**

Functions available for Presentation Management are:

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

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

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

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

— Manipulate window cursors (create, destroy, assign, change, etc.)

— Draw text and obtain text metric information

**Event Handling**

The basic window services support application requirements to respond to the human user's actions, rather than forcing the human 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, etc.)

— Window management events (window exposure event, colormap event, etc.)

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

Functions available for Event Handling are:

— Select events

— Map events to a window

— Get information about events

— Send events

**Error Handling**

Functions available for Error Handling are:

— Get error message

— Get error description

— Set error event handler routine

**Interclient Communication**

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

— Properties

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

— Selections

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

— Cut Buffers

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

Functions available for Interclient Communication are:

— Manipulate window properties (list, delete, change, get, etc.)

— Set and get selections

— Manipulate cut buffers

**Input Device Management**

Functions available for Input Device Management are:

— Receive keyboard input and pointing device button events

— Gain exclusive control of keyboard or pointing cursor

— Track the pointing cursor

— Change Server-wide keyboard mappings

— Set and get keyboard and pointing device preferences

**Screen Management**

Functions available for Screen Management are:

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

— 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, etc.)

**User Preferences Management**

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

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

— The human 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, etc.]

— 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

#### 4.6.4.1.2 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 human users (typically via a keyboard and a pointing device) and communicates with applications and other visual object software. The graphical representation of a visual object can be modified to reflect the results of application processing. Examples of visual objects are graphical push buttons, check boxes, and editing boxes. (Note: The term used within the X Window System community to define visual objects is ''widgets.'')

Toolkit Window services are provided for two reasons:

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

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

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

An application may use Toolkit Window services to:

— Perform toolkit initialization/exit

— Set up visual object resources

— Create/delete a visual object

— Display a visual object

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

— Retrieve/modify the state of a visual object

— Turn control over to the toolkit for user input processing

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

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

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

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

— Handle user input

— Manage visual object resources

— Translate an event into an action

— Manipulate graphics contexts

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

— Manage memory associated with HCI

— Handle errors associated with HCI

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

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

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

#### 4.6.4.1.3 Dialog Services

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

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

These services are subdivided into:

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

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

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

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

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

#### 4.6.4.2 External Environment Interface Services

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

— HCI : provides definition of the presentation and behavior of the visual display objects, command language definition (syntax and semantics), specifications related to keyboards, selection devices, audio and video input/output devices.

— Information Interfaces: provides specification of user resource data formats, containing presentation and action information pertaining to visual display objects.

— Network Interfaces: provides protocol services for data transport, which is basically the bottom six layers of the OSI model

#### 4.6.4.3 Interapplication Entity Services

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

#### 4.6.4.4 Windowing Resource Management Services

These services provide general management functions across the HCI components, which include system administration-oriented functions (i.e., management of human/computer interfaces within the scope of the administrator, such as setting up defaults and human user customization functions. For instance, it is important to allow reconfiguration and addition of terminals and displays without affecting the application interface.) These resource management services are independent from the OLTP Resource Management Services defined in 4.10.4.4.

### 4.6.5 Standards, Specifications, and Gaps

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

#### 4.6.5.1 Current Standards in the POSIX OSE

No current international or national standards exist for the HCI Services, primarily due to the recent emergence of the windowing technology. However, several standard activities are underway and referenced under 4.6.5.2.

There is a de facto standard for the base window system. The X Window System windowing protocol and the Xlib functional interface to the protocol were