Common Facilities Architecture

Revision 4.0
November 1995

NOTICE

The information contained in this document is subject to change without notice.

The material in this document details an Object Management Group specification in accordance with the license and notices set forth on this page. This document does not represent a commitment to implement any portion of this specification in any companies' products.

WHILE THE INFORMATION IN THIS PUBLICATION IS BELIEVED TO BE ACCURATE, THE OBJECT MANAGEMENT GROUP AND THE COMPANIES LISTED ABOVE MAKE NO WARRANTY OF ANY KIND WITH REGARD TO THIS MATERIAL INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. The Object Management Group and the companies listed above shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance or use of this material.

Table of Contents

Preface

0.1 Document Structure

CORBAfacilities: Common Facilities Architecture is organized as follows:

Chapter 1, “Overview” positions the Common Facilities Architecture with the other work of the Object Management Group and provides an overview of the Common Facilities Architecture.

Chapters 2 through 5 describe each of the Horizontal Common Facilities: user interface, information management, systems management, and task management. The Horizontal Common Facilities are those facilities that are used by most systems.

Chapter 6, “Vertical Market Facilities” describes the Vertical Market Facilities, which are those facilities that are specific to particular domains or industries, rather than widely applicable.

Appendix A describes the Internationalization and Security Facilities, which are specializations of the Internationalization and Security Object Services.

Appendix B contains a glossary.

Finally, this document contains a list of the reference material used to write *CORBAfacilities*.

0.2 About This Document

Under the terms of the collaboration between OMG and X/Open Co Ltd, this document is a candidate for endorsement by X/Open, initially as a Preliminary Specification and later as a full CAE Specification. The collaboration between OMG and X/Open Co Ltd ensures joint review and cohesive support for emerging object-based specifications.

X/Open Preliminary Specifications undergo close scrutiny through a review process at X/Open before publication and are inherently stable specifications. Upgrade to full CAE Specification, after a reasonable interval, takes place following further review by X/Open. This further review considers the implementation experience of members and the full implications of conformance and branding.

0.2.1 Object Management Group

The Object Management Group, Inc. (OMG) is an international organization supported by over 700 members, including information system vendors, software developers and users. Founded in 1989, the OMG promotes the theory and practice of object-oriented technology in software development. The organization's charter includes the establishment of industry guidelines and object management specifications to provide a common framework for application development. Primary goals are the reusability, portability, and interoperability of object-based software in distributed, heterogeneous environments. Conformance to these specifications will make it possible to develop a heterogeneous applications environment across all major hardware platforms and operating systems.

The OMG's objectives are to foster the growth of object technology and influence its direction by establishing the Object Management Architecture (OMA). The OMA provides the conceptual infrastructure upon which all OMG specifications are based.

0.2.2 X/Open

X/Open is an independent, worldwide, open systems organization supported by most of the world's largest information system suppliers, user organizations and software companies. Its mission is to bring to users greater value from computing, through the practical implementation of open systems. X/Open’s strategy for achieving its mission is to combine existing and emerging standards into a comprehensive, integrated systems environment called the Common Applications Environment (CAE).

The components of the CAE are defined in X/Open CAE specifications. These contain, among other things, an evolving portfolio of practical application programming interfaces (APIs), which significantly enhance portability of application programs at the source code level. The APIs also enahance the interoperability of applications by providing definitions of, and references to, protocols and protocol profiles.

The X/Open specifications are also supported by an extensive set of conformance tests and by the X/Open trademark (XPG brand), which is licensed by X/Open and is carried only on products that comply with the CAE specifications.

0.3 Purpose of this Document

The Common Facilities architecture described in this document is a management tool to direct the adoption of specifications for Common Facilities. It should used as a guideline in the preparation and evaluation of Requests for Proposals and Requests for Comments.

The objectives for this document are to:

- Identify basic Common Facilities of interest to the OMG community
- Position Common Facilities within the Object Management Architecture
- Describe dependencies among Common Facilities

The *Common Facilities Architecture* will be revised as Common Facilities specifications are adopted by the OMG and as requirements for Common Facilities evolve.

0.4 Intended Audience

The architecture described in this manual is aimed at managers and software designers who want to produce applications that comply with the family of OMG standards. The benefit of compliance is, in general, to be able to produce interoperable applications that run in heterogeneous, distributed environments.

0.5 Associated Documents

The CORBA documentation set includes the following books:

- *Object Management Architecture Guide* defines the OMG's technical objectives and terminology and describes the conceptual models upon which OMG standards are based. It defines the umbrella architecture for the OMG standards. It also provides information about the policies and procedures of OMG, such as how standards are proposed, evaluated, and accepted.
- *CORBA: Common Object Request Broker Architecture and Specification* contains the architecture and specifications for the Object Request Broker.
- *CORBAservices: Common Object Services Specification* contains specifications for OMG's Object Services.
- *CORBAfacilities: Common Facilities Architecture* describes an architecture for Common Facilites. Over time, specifications for Common Facilities, based on this architecture, will be adopted and published by the OMG.

The OMG collects information for each book in the documentation set by issuing Requests for Information, Requests for Proposals, and Requests for Comment and, with its membership, evaluating the responses. Specifications are adopted as standards only when representatives of the OMG membership accept them as such by vote. (The policies and procedures of the OMG are described in detail in the *Object Management Architecture Guide*.)

To obtain books in the documentation set or other OMG publications, refer to the enclosed subscription card or contact the Object Management Group, Inc. at:

OMG Headquarters
492 Old Connecticut Path
Framingham, MA 01701
USA
Tel: +1-508-820 4300
Fax: +1-508-820 4303
pubs@omg.org
http://www.omg.org/

For more information about obtaining and responding to RFIs, RFPs, and RFCs for Common Facilities, contact the OMG.

0.6 Context of Common Facilities

The OMG is dedicated to producing a framework and specifications for commercially available object-oriented environments. The Object Management Architecture (as defined in the *Object Management Architecture Guide*) is the umbrella architecture for OMG specifications. The defining model for the architecture is the Reference Model, which classifies the components, interfaces, and protocols that compose an object system. The Reference Model consists of the following components:

- **Object Request Broker**, which enables objects to transparently make and receive requests and responses in a distributed environment. It is the foundation for building applications from distributed objects and for interoperability between applications in hetero- and homogeneous environments. The architecture and specifications of the Object Request Broker are described in *CORBA: Common Object Request Broker Architecture and Specification*

- **Object Services**, a collection of services (interfaces and objects) that support basic functions for using and implementing objects. Services are necessary to construct any distributed application and are always independent of application domains. For example, the Life Cycle Service defines conventions for creating, deleting, copying, and moving objects; it does not dictate how the objects are implemented in an application. Specifications for Object Services are contained in *CORBAservices: Common Object Services Specification.*

- **Common Facilities**, a collection of services that many applications may share, but which are not as fundamental as the Object Services. For instance, a system management or electronic mail facility could be classified as a common facility. Common Facilities are divided into two major categories: Horizontal Common Facilities, which are used by most systems, and Vertical Market Facilities, which are domain-specific. Information about the architecture of Common Facilities is contained in this manual.

- **Application Objects**, which are objects specific to particular commercial products or end user systems. Application Objects correspond to the traditional notion of applications, so they are not standardized by the OMG. Instead, Application Objects constitute the uppermost layer of the Reference Model.

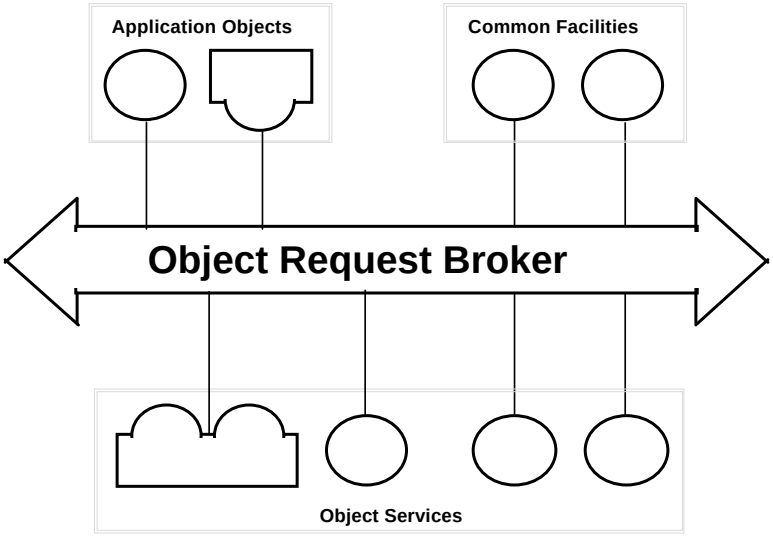

Figure 0-1 Reference Model: Context of Common Facilities

0.7 Acknowledgments

The following individuals provided significant contributions to this document:

Apple Computer, Inc.	Borek Vokach-Brodsky
Component Integration Labs	Jed Harris, Neil Katin
Crystaliz, Inc.	Sankar Virdhagriswaran
Hughes Applied Information Systems, Inc.	Tom Herron
IBM Corporation	David Zenie; Kathy Bohrer; Todd Scallan
ICL	Andrew Hutt
MITRE Corporation	Chris Irwin; Melony Katz; Fred Kuhl; Beth Lavender; Tom Mowbray; Diane Mularz

Novell, Inc.	William Cox
Petrotechnical Open Software Corp.	Alan Doniger
Schlumberger Technologies	Claude Baudion
SEMATECH Inc.	Fred Waskhewicz
SunSoft, Inc.	Geoff Lewis, Neil Katin
Tandem Computers, Inc.	Kent Salmond
WordPerfect Corporation	Marc Ericson
X/Open Co. Ltd.	Martin Kirk

Overview 1

This chapter introduces the Common Facilities. It provides the following information:

- An explanation of how Common Facilities fit into the OMG's umbrella architecture, the Object Management Architecture.
- Tables that list the Common Facilities that have been identified by the OMG.
- A description of the template that the OMG used to describe each Common Facility in this book, and that can be used to describe future facilities.

1.1 Role of Common Facilities in the OMA

This section addresses the role of Common Facilities in the Object Management Architecture (OMA), so it describes Common Facilities as they pertain to the Object Request Broker, Object Services, and Application Objects.

The initial focus of the OMG effort was the Object Request Broker (ORB). The ORB provides the basic communication channel through which objects interact to provide system services. Since all object behavior is defined in terms of messages exchanged among objects, the communication protocol defined by the ORB is in effect the grammar for all other OMA specifications.

While the ORB specifies a system's grammar, Object Services represent its most basic vocabulary: the essential interfaces needed to create an object, introduce it into its environment, use and modify its features, and so forth. These services, bundled with every ORB, constitute the basic enabling technology of an OMA-compliant software system.

Common Facilities is the final area of the Object Management Architecture to be defined. They fill the conceptual space between the enabling technology defined by CORBA and the Object Services, and the application-specific (and hence by definition unstandardized) services that the OMA labels "Application Objects."

Common Facilities include specifications for higher level services and vertical market speciality areas. Some general purpose examples of Common Facilities include email, printing, and compound documents. These types of Common Facilities are needed in most application domains. In addition, there are many OMG-related groups working on more specialized Common Facilities, such as geo-spatial data processing and system management. Common Facilities is an appropriate area for standards providing interoperability between independent software vendors' (ISV) products.

Common Facilities are separated into two categories:

- **Horizontal Common Facilities**, which are shared by many or most systems. There are four major sets of these facilities: User Interface, Information Management, Systems Management and Task Management.
- **Vertical Market Facilities**, which support the domain-specific tasks that are associated with vertical market segments.

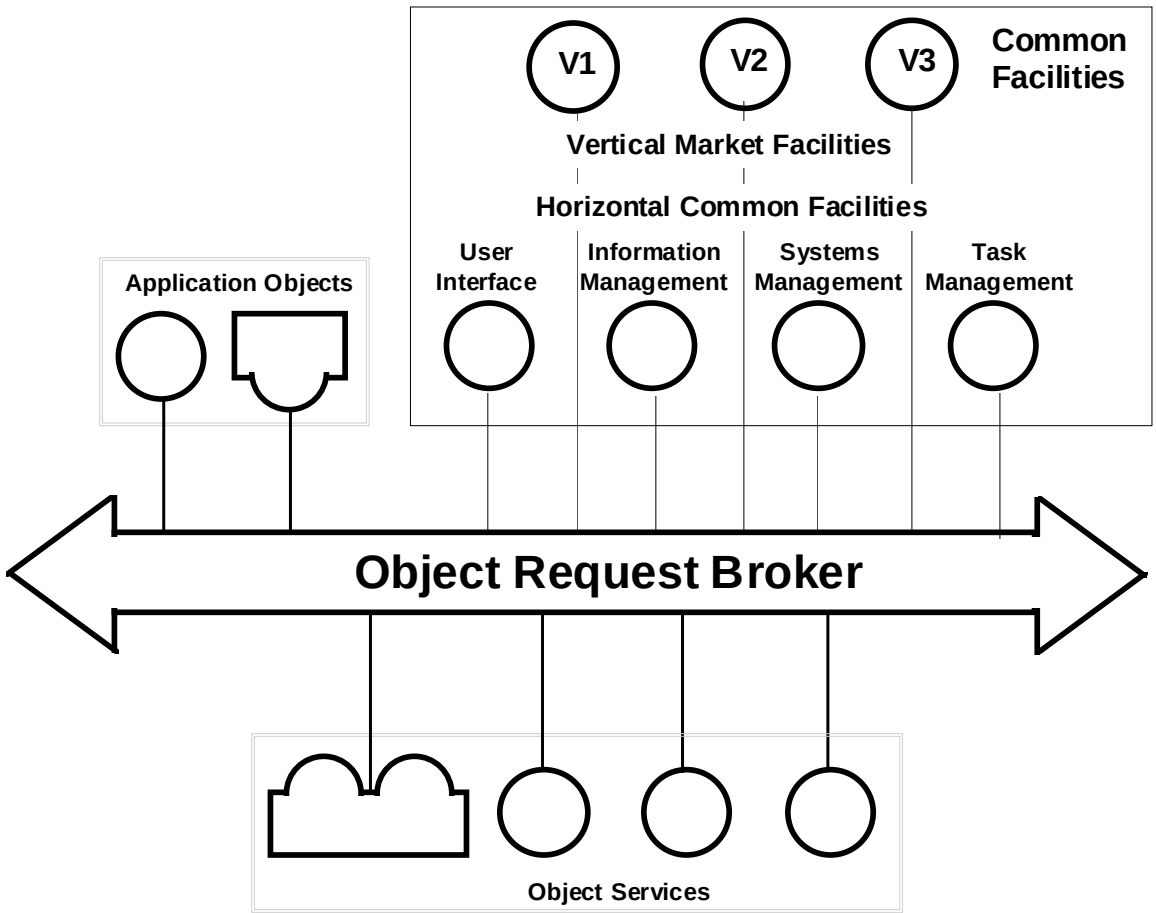

Figure 1-1 Common Facilities in the OMA.

1.1.1 Evolution of Common Facilities

Some Vertical Market Facilities may migrate to Horizontal Common Facilities. Services that are common across many vertical facilities areas are candidates for horizontal facility status.

The boundaries separating Common Facilities from Application Objects and from Object Services are thus not hard and fast, but reflect the evolution of object system technology. The current placement of the boundaries reflects the current OMG standardization effort. As experience in an application area matures, areas of potential new Common Facilities will be discovered and defined, just as evolving system infrastructures will gradually incorporate pieces of the Common Facilities domain into their basic Object Service offerings.

Operations provided by Object Services are expected to serve as building blocks which are used by Common Facilities and Application Objects. Common Facilities provide higher level interoperable interfaces for Application Objects which can specialized for specific application domains. Extensive inheritance relationships should exist between Object Services, Common Facilities, and Application Objects, as shown in Figure 1-2 on page 1-3. Inheritance will facilitate the reuse of standard interfaces, interoperability between objects conforming to the base standard, and increased consistency of interface design between object types.

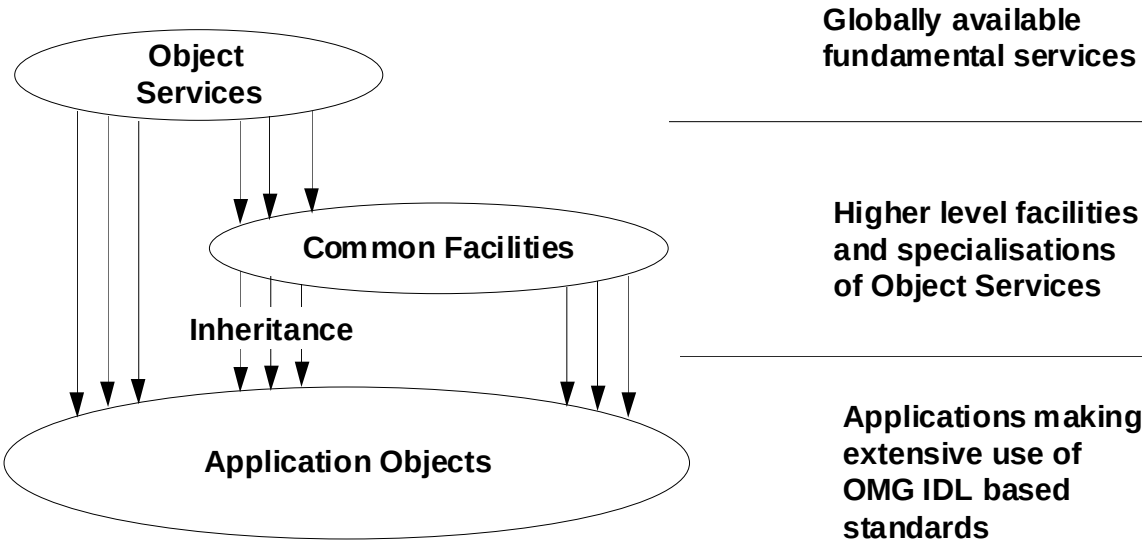

Figure 1-2 Reuse of Object Services and Common Facilities

In non-object software systems, a system's application program interface (API) often is defined by a monolithic interface. The Common Facilities API is modular; particular objects may use a few or many Common Facilities. By being object-oriented, the Common Facilities API is extensible, customizable, and subsettable; applications only need to use facilities they require.

The operations provided by Common Facilities are made available through the OMG Interface Definition Language (OMG IDL, as defined in *CORBA: Common Object Request Broker Architecture and Specification*) or through proposed extensions to OMG IDL. Extensions must be compatible with the OMG Object Model, which is described in the *Object Management Architecture Guide*.

Although OMG requires an OMG IDL interface for each Common Facility, it is worth noting that since OMG adopts existing technology, implementations of Common Facilities may not themselves be object-oriented. In addition to an OMG IDL interface, non-object-oriented interfaces may continue to be supported for compatibility with an existing product's API or with non-OMG standards. Such interfaces will not, however, be part of OMG specifications. Also note that objects do not have to use the implementation of basic operations provided by Common Facilities nor do objects have to provide all basic operations. For example, an object might provide its own data storage; an object that models a process might not provide transactions.

1.1.2 Mapping Common Facilities to Cargill's Model

The role of Common Facilities can be defined in terms of Cargill's published model of information technology standardization. (See Figure 1-3 on page 1-5). In the model, the Object Management Architecture provides a generic, long term reference model for the OMG standards. CORBA and Object Services are globally applicable industry standards that address the universal needs of applications. Looking at the application use of standards, Cargill identifies functional profiles as specializations of industry standards, systems profiles as further specializations that are generalizations across multiple applications, and application implementations.

Common Facilities include some industry standards and some first-level functional profiles. The Horizontal Common Facilities are industry standards that provide for general purpose horizontal market needs. The Vertical Market Facilities are functional profiles that address industry needs within specific vertical market areas. Application Objects in the OMA architecture comprise Cargill's second level functional profiles,

systems profiles, and application implementations. Eventually, there will be a need for an object type registry so that Application Object specifications can be registered and shared.

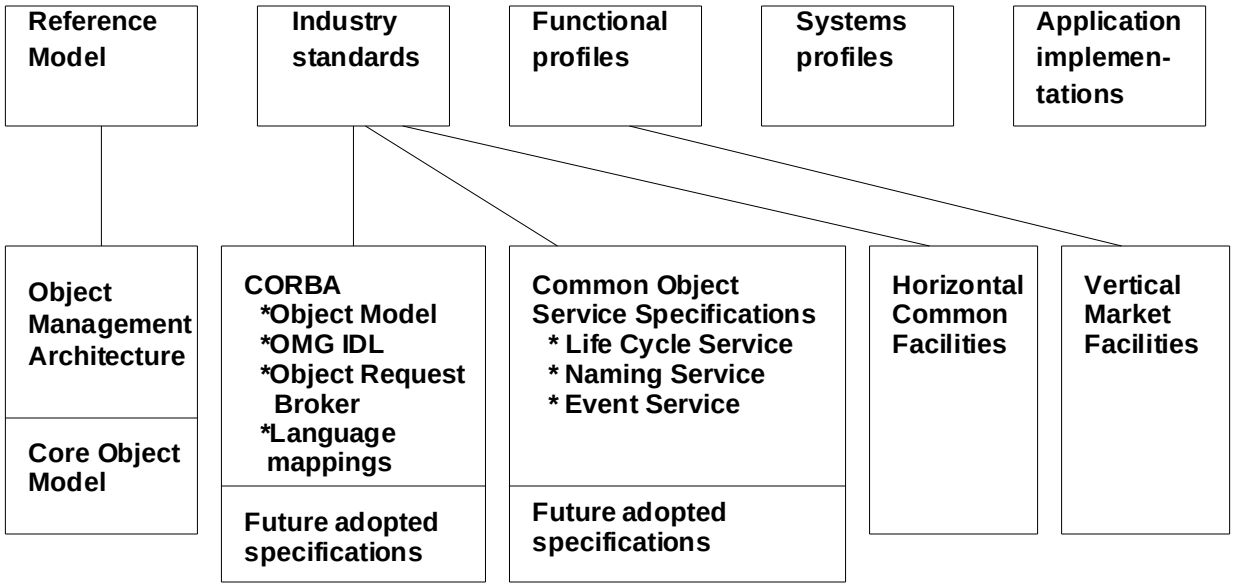

Figure 1-3 Cargill's Model of Technology Standardization Applied to Common Facilities

1.2 Description of Common Facilities

As shown in Figure 1-1 on page 1-2, there are two major categories of Common Facilities: Horizontal Common Facilities and Vertical Market Facilities. The OMG has developed a high level model for each of the Horizontal Common Facilities and a list of components that fit into that model. The OMG has also developed a list of Vertical Market Facilities.

1.2.1 Horizontal Common Facilties

Horizontal Common Facilities include functions shared by many or most systems, regardless of application content. Four major domains for such facilities have been identified so far:

- **User Interface** makes an information system accessible to its users and responsive to their needs. For example, the Compound Presentation Facility is a facility in the user interface domain.

- **Information Management** covers the modeling, definition, storage, retrieval, management, and interchange of information. The Compound Interchange Facility is an element of the information management domain.
- **System Management** covers the management of complex, multi-vendor information systems by service providers.
- **Task Management** covers the automation of work. This includes automation of both user processes and system processes which operate as part of the information system.

User Interface Common Facilities

The facilities listed in Table 1-1 on page 1-6 are described in detail in Chapter 2, “User Interface Common Facilities”.

Table 1-1

User Interface Common Facilities	Capabilities
Rendering Management	Supports general purpose presentation of objects, such as printing and display.
Compound Presentation Management	Supports the presentation (e.g printing and display) of objects in compound documents.
User Support Facilities	Mechanism for storing and presenting application help information and for handling common requirements such as text checking.
Desktop Management	Provides facilities for the end user desktop.
Scripting	Supports the interactive creation of automation scripts.

Information Management Common Facilities

The facilities listed in Table 1-2 on page 1-6 are described in detail in Chapter 3, “Information Management Common Facilities”.

Table 1-2

Information Management Common Facilities	Capabilities
Information Modeling	Supports the creation of information models and schemas.
Information Storage and Retrieval Facility	Supports the persistent storage of information and its retrieval.
Compound Interchange	Supports the interchange of data in compound documents.
Data Interchange	Supports the general interchange of data.
Information Exchange	Supports the interchange of information.

Table 1-2

Data Encoding and Representation	Supports facilities for data format encodings and translations.
Time Operations	Supports facilities for the manipulation of calendar and time data.

System Management Common Facilities

The facilities listed in Table 1-3 on page 1-7 are described in detail in Chapter 4, “System Management Common Facilities”.

Table 1-3

System Management Common Facilities	**Capabilities**
Management Tools	Supports the interoperability of management tools and collection management facilities.
Collection Management	Supports the integration of collection management facilities and managed objects.
Control	Supports the control of system resources and managed objects.

Task Management Common Facilities

The facilities listed in Table 1-4 on page 1-7 are described in detail in Chapter 5, “Task Management Common Facilities”.

Table 1-4

Task Management Common Facilities	**Capabilities**
Workflow	Provides management and coordination of object that are part of a work process.
Agent	Provides support for static and dynamic agents.
Rule Management	Supports the knowledge acquisition, maintenance, and execution of rule based objects, such as intelligent agents.
Automation	Allows access to the key functionality of one object from another object.

1.2.2 Veritcal Market Facilities

The Vertical Market Facilities represent technology that supports various vertical market segments such as health care, retailing, manufacturing, or financial systems. This is by far the larger set of facilities. As industry groups develop such facilities, the OMG will help them to integrate their efforts into the Common Facilities architecture.

The facilities listed in Table 1-5 on page 1-8 are described in more detail in Chapter 6, “Vertical Market Facilities”.

Table 1-5

Vertical Market Facilities	Capabilities
Imagery	Provides interoperability between imagery objects, image related information, and imagery application services.
Information Superhighways	Supports multi-user information service applications across wide area networks.
Manufacturing	Supports interoperability between manufacturing objects.
Distributed Simulation	Supports the interaction of multiple simulation objects in virtual environments.
Oil and Gas Industry Exploitation and Production	Supports interoperability in the petroleum vertical market.
Accounting	Supports commercial transactions.
Application Development	Supports interoperability between application development objects.
Mapping	Supports interoperability between mapping objects.

1.2.3 Template for Describing Common Facilities

Each of the chapters that follow describes a set of Common Facilities according to a template.

There are two levels of description: high level and detailed. The high level description provides a overview model of the set of facilities. This model helps to scope the set and to describe how this set relates to other sets. Detailed descriptions of individual Common Facilities are provided to give general guidance on how future work may proceed.

At the detailed level, each Common Facility description conforms to the following template:

Description and Requirements describes a common facility and delimits the scope of the facility. The description is written as a collection of requirements and goals.

Related Standards and References lists documents that could be useful to submitters in understanding the problem space. Submitters should describe whether and how their proposals relate to existing, relevant standards.

Relationship to Components of OMA describes possible interactions with or relationships to other Common Facilities, Object Services, CORBA, and the OMG Object Model. In particular, it describes:

- Other Common Facilities*:* How the facility depends on the existence of other facilities.

- Object Services: How the facility depends upon the Object Services and how the service may use or specialize the *CORBAservices* specification.

- CORBA: How the facility may use features described in the *CORBA* document.

- OMG Object Model: How the service may conform to the Object Model (as defined in *Object Management Architecture Guide*) and what new components or profiles may be needed.

Interactions among Common Facilities are described as follows:

- Interactions between the sets of Common Facilities are described in the overview sections.

- Interactions between a given Common Facility and other facilities, Object Services and with CORBA are described in the detailed description of the facility.

Technical Issues provides a log of additional technical issues that the Common Facilities Architecture does not address. "None" is a potential response for this section.

User Interface Common Facilities 2

2.1 Overview

The User Interface facilities cover all aspects of the user interface. They include the enablers used to deliver a user interface and the tools used by application developers to develop that interface. It also includes the facilities needed to give users easy access to their applications and to automate parts of their work.

Figure 2-1 on page 2-2 illustrates the components of a user interface. The components are as follows:

The user interface style defines the look and feel of the user interface in terms of user interface objects (for example, a menu bar) and the actions users can perform on these objects. It also prescribes the use of the workstation hardware.

The workstation hardware includes the equipment used to deliver the user interface, that is screen, keyboard, mouse, printer and security devices. It is described in terms of its functional capability and also its physical properties (for example, screen luminance and keyboard layout).

User interface enablers deliver the user interface to a range of applications. There are several types of enabler, including window managers, terminal emulator programs, and class libraries of user interface objects.

User interface enablers are grouped into three main Common Facilities:

- Rendering management facility, which provides access to and abstractions of the user interface hardware, support for windows, user interface objects and dialogue objects.

- Compound presentation facility, which provides a framework for subdividing a display window into multiple parts and maps the display portion of a compound document into them.

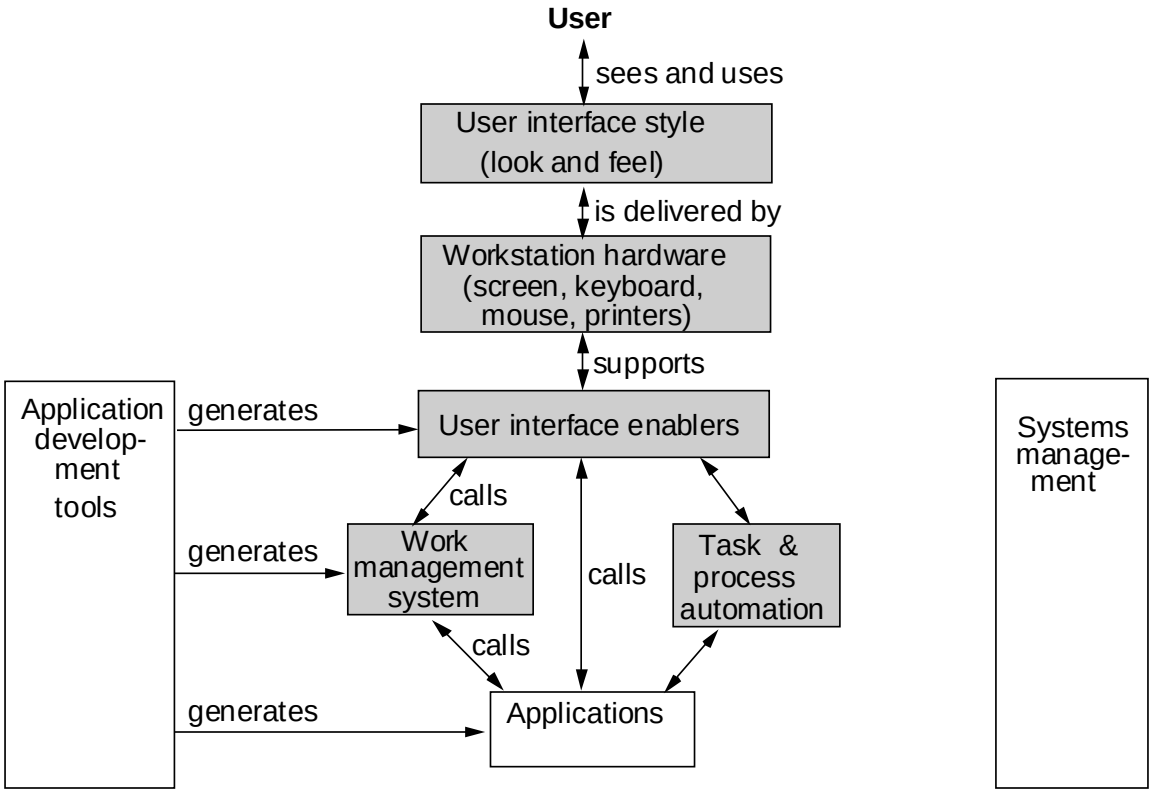

Figure 2-1 Components of the User Interface Facility

- User support facilities, which provide facilities that are common across applications such as help and text checking.

The work management system enables an information system to maintain the user's working context. The capabilities of this component include:

- Support for user single system log-on.
- Definition of the user's working environment in terms of the applications and information used daily.
- Desktop management which provides iconic desktops as a means of visualising the user's working environment.

Task and process automation provides facilities to allow users to:

- Automate their tasks through the use of scripting.
- Automate desk procedures by writing descriptions of those procedures in a simple language.
- Participate in enterprise business procedures by using workflows.

2.2 Rendering Management

2.2.1 Description and Requirements

Rendering Management provides facilities to present information for output on devices such as screens, printers, plotters and sound and speech output devices. It also provides facilities to handle user input from a variety of different hardware devices such a key-board, mouse, scanners, speech recognition devices, and security devices.

Rendering Management includes support for:

- Window management.
- Class libraries for user interface objects.
- User interface dialogue objects (for example, menu bars, scroll bars, and so forth).
- Abstractions of the many different input and output devices.

2.2.2 Related Standards and References

Many user interface styles define the user interface objects that need to be supported by rendering management. These styles include:

Apple Computer, *Macintosh Human Interface Guidelines*, Addison-Wesley Publishing Company, Reading, MA 1992.

International Business Machines (IBM), *Systems Application Architecture, Common User Access Guide to User Interface Design*, October 1991.

International Business Machines (IBM), *Systems Application Architecture, Common User Access Advanced Interface Design Reference*, October 1991.

Microsoft Corporation, *The Windows Interface: An Application Style Guide*, Microsoft Press, Redmond WA 1992.

Microsoft Corporation, *The Microsoft Windows User Interface Design Guide* (draft), 1992.

NeXT Computer Inc., *NEXTSTEP User Interface Guidelines*, November 1993.

NeXT Computer Inc., *NEXTSTEP General Reference, Volume 1*, November 1993.

Open Software Foundation, *OSF/Motif Style Guide (Release 1.2)*, Prentice-Hall, Inc., Englewood Cliffs, NJ 1993.

Sun Microsystems, Inc. *OPEN LOOK Graphical User Interface Application Style Guidelines*, Addison-Wesley Publishing Company, Reading, MA 1990.

Sun Microsystems, Inc. *OPEN LOOK Graphical User Interface Functional Specification Guidelines*, Addison-Wesley Publishing Company, Reading, MA 1989.

2.2.3 Relationship to Other Components of OMA

Rendering management provides a vital link between the hardware and operating system and the Compound Presentation Facility.

2.2.4 Technical Issues

None.

2.3 Compound Presentation Facility

2.3.1 Description and Requirements

The Compound Presentation Facility should provide a framework for sharing and subdividing a display window into multiple parts. These parts may be peers of each other or may in turn may be embedded into other parts. This facility maps to the display portion of a compound document architecture.

There are many issues to be considered in the Compound Presentation facility. The facility should address:

- Geometry management.
- Human interface event distribution.
- Shared human interface control management (for example, menus, palettes, button bars).
- Rendering management (including printing).

2.3.2 Relationship to Components of OMA

None.

2.3.3 Related Standards and References

None.

2.4 User Support Facility

2.4.1 Description and Requirements

Today, a number of facilities and capabilities are provided to the end user by individual applications that are duplicated across applications and provide similar functionality. Although there may be a set of general requirements that are shared among these non-application function specific facilities, they have typically been designed and implemented on a custom basis. This diversity in development results in different presentation schemes, styles and semantics to the user. These facilities also become a static part of the application, increasing the code and data size.

Providing these non-application specific functions as stand-alone, reusable Common Facilities provides advantages to the user and the developer. The user will benefit from consistent interface style, semantics, and predictable behavior. The developer will benefit from reusable code with standardized interfaces.

The number and function of User Support facilities is expected to expand over time as more application logic is abstracted into reusable components. In the future, User Support Facilities will cover the following cross application functions:

- Help
- Text Checking

In the future, User Support facilities will expand to include versioning; annotating; standard text; graph and spreadsheet functions; and other, additional facilities.

Help User Support Facility

The Help system must be stand-alone and provide mechanisms to access, present, and interchange help data, including internationalized text; image; sound; and animation. It must also support multiple presentation styles (such as context-sensitive, a manual browser, and hyper-text/media); a standard storage format (for example, SGML); and multiple storage styles (for example, all help data in a single file, a file per page of display, and so forth).

The Help Facility must include OMG IDL specifications for at least the following interfaces:

- Initializing and freeing the Help Facility object.

- Attaching (and detaching) help objects to any objects in the application so that appropriate help data can be rendered (and nullifying the rendering operation) when the user requests help. This operation may need to take a marker name if the help file contains multiple segments of help data. Time-based data such as sound and animation may need begin-time and end-time information if only a portion of data in the file is to be rendered. An optional parameter may also be needed to indicate the name of other applications that need to be invoked.

- Rendering help objects for programmatically displaying the help data. The parameters are the same as in attaching and detaching.

- Querying for help event history. There should be a top-level help object that manages the sub-components, interaction with other objects, and interface to other OMA components such as event management. If the current help is rendered in response to a user action from another help object (i.e. nested), the information of the level of nesting should be made available to the application on request.

- Querying the current help file path and name.

- Querying the object name to which it is attached.

- Printing to send the currently rendered help data to a hard copy service.

Text Checking User Support Facility

The Text Checking Facility must be stand-alone and provide mechanisms to allow applications to pass strings or files of internationalized text to be subjected to various lexical checks, including, at a minimum, spelling, hyphenation, thesaurus and grammar. The facility must provide the capability to support text strings independent of font and mark-up symbols, and must be able to interact with the application in order to check a single string of text, or an entire file or document.

This facility must include OMG IDL specifications for at least the following interfaces:

- Initializing and freeing the text checking objects.

- Sending and receiving text strings with both the original text and the changed text (to support versioning in the application).

- Provide customization of the service to be executed against the string, including at least: spelling check; hyphenation check; thesaurus; and grammar check.

- Provide capability to query the dictionary in use, and to provide alternative, application-provided dictionaries.

2.4.2 Related Standards and References

ISO 9241 *Ergonomic Requirements for Office Work with Visual Display Terminals, Part 13 User Guidance* (not yet published).

ISO/IEC 8879:1986 Standardized Generalized Markup Language (SGML).

IEEE P1201.2 *Recommended Practice for Graphical User Interface Drivability*, Balloting Draft 2, August 1993.

Human Factors Society (HFS) Human-Computer Interaction (HCI) Committee, *User Guidance* (draft), April 1991.

Apple Computer, *Macintosh Human Interface Guidelines*, Addison-Wesley Publishing Company, Reading, MA 1992.

International Business Machines (IBM), *Systems Application Architecture, Common User Access Guide to User Interface Design*, October 1991.

International Business Machines (IBM), *Systems Application Architecture, Common User Access Advanced Interface Design Reference*, October 1991.

Microsoft Corporation, *The Windows Interface: An Application Style Guide*, Microsoft Press, Redmond WA 1992.

Microsoft Corporation, *The Microsoft Windows User Interface Design Guide* (draft), 1992.

NeXT Computer Inc., *NEXTSTEP User Interface Guidelines*, November 1993.

NeXT Computer Inc., *NEXTSTEP General Reference, Volume 1*, November 1993.

Open Software Foundation, *OSF/Motif Style Guide (Release 1.2)*, Prentice-Hall, Inc. Englewood Cliffs, NJ 1993.

Sun Microsystems, Inc. *OPEN LOOK Graphical User Interface Application Style Guidelines*, Addison-Wesley Publishing Company, Reading, MA 1990.

Sun Microsystems, Inc. *OPEN LOOK Graphical User Interface Functional Specification Guidelines*, Addison-Wesley Publishing Company, Reading, MA 1989.

2.4.3 Relationship to Components of OMA

The User Support facilities should require no changes to the other OMG standards. The Facilities should use the OMG Object Management Architecture, Object Model, Common Object Request Broker Architecture, Object Services, and other Common Facilities as specified. In particular, the User Support facilities will benefit from other Common Facilities such as User Interface facilities and the Information Management facilities, and may

use and specialize Object Services such as the Event Service. (The Event Service is described in *CORBAservices*.)

2.4.4 Technical Issues

None.

2.5 Desktop Management Facility

2.5.1 Description and Requirements

Desktop Management facilities provide a general purpose infrastructure for visualizing the user's working environment. This infrastructure supports a conceptual model of user objects that is specialized by desktop implementations that support further specialization and instantiation by users, projects, and enterprises.

The fundamental types of user objects are:

Information: objects that are processed by tools in the context of user tasks. An information object can be associated with a workflow, which defines the operations and policies which can be applied to the information. This workflow may be may be composed of a single operation or many serial and parallel operations sequenced by rules. Information objects may contain application information as well as workflow, model, resource, and project information. Information objects may represent or be part of:

- Aggregations, which group together information in n-level container hierarchies
- Versions, which represent information evolution
- Configurations, which represent usage or consistency

Tools: objects that operate on user information. There are application tools such as editors and simulators, desktop tools such as browsers and workflow editors, system tools for managing operating systems, and hardware tools such as printers.

Tasks: objects that express the context in which tools process information. Tasks are instances of workflows bound to instances of information. Users are guided through tasks by workflow rules that determine the next operation, or operations, to execute based on interpretation of task context and state.

Access to user objects, in an arbitrarily large and heterogenous, multi-user environment, is restricted by security and concurrency mechanisms. The appearence and behavior is consistent regardless of the scale and complexity of the network containing, and facilities managing, user objects. For example, information objects which are distributed and have relationships that are being navigated or established across multiple heterogenous information management facilities, behave on the desktop as if they were managed by a single facility.

Desktop Management facilities support collaboration in formally defined projects and in informal transactions by managing user transactions that generate requests to appropriate information, task, and system management facilities. The facilities are responsible for:

- Defining user profiles, organization and project structures that specify and assign group and role membership used by facilities to determine resource access.
- Synchronization and concurrency mechanisms such as locking and versioning that facilitate and provide integrity for parallel activities.

Desktop Management facilities support the following functions:

- Installation and setup
- Session management
- Information management
- Tool management
- Task management

Installation and Setup

Installation and setup is required for for new installations, extensions, and modifications to the user's working environment. Such installations may include a network of heterogenous facilities and platforms. Operating systems and facilities may be installed independently of this function; however, their existence and context is registered through interaction with this function.

The installation and setup process should be automatic except where information or choices are required. The dialogue with the system administrator executing this process must setting of environment variables, configuration files, and other internal information is the responsibility of this function based on information obtained from dialogue with the system administrator.

Session Management

The session management function is required for connection, customization, control, and recovery of the user's working environment, including:

- Login, which includes use of security services and the handling of exception and other events that have occurred since the last connection to establish and update the user's working environment.
- Logout, which includes support for handling uncommitted data and executing tools that are unable to continue execution while the user is not connected.

- System Configuration, which provides support for customizing and extending the user's working environment from an installed desktop.
- Event Logging, providing support for viewing and setting retention and usage policies.
- Undo/Redo, to support for reversing user transactions.
- Service functions for fault isolation, detection, and correction at the desktop.
- User, Group, and Role functions for creation, definition, and assignment.

Information Management

The information management function must support interfacility relationships, transactions, and notifications. Information management functions are presented in separate windows that inherit the behavior of these common functions. The desktop provides a consistent user interface to information management facility functions and assists with interfacility cooperation. The common functions are:

- Creation, destruction and concurrent usage (checkout/in) of objects.
- Browsing, navigation, and creation/destruction of object relationships.
- Functions for registering interest in object change notifications, setting and querying attributes, defining information types, and backup/restore.

Tool *Management*

Tool management functions provide the user interface for defining, registering, and using tools managed by tool management facilities. This includes:

- Editors for specifying tool characteristics and execution requirements.
- Methods for constructing and using collections of tools in toolboxes.
- Execution of selected tools with interpretation of conditions specifying information required, tool version, access, and execution environment.

Task Management

Task management functions provide the user interface for task creation, execution, control, and resource management. Creation involves binding an object to a workflow, or to a tool, with defaults presented based on information type. Execution, control, and scheduling functions include start, suspend, and resume task.

2.5.2 Related Standards and References

CAD Framework Initiative (CFI) Task and Session Model, May 2 1994.

ICL Common Facilities, Response to OMG Common Facilities Request For Information.

Inter-Framework Facility, Response to OMG Common Facilities Request For Information.

2.5.3 Relationship to Other Components of OMA

The Desktop Management Facility requires a protocol that supports cooperation between Task Management, Information Management and System Management Common Facilities for issuing requests based on desktop transactions.

The Desktop Management Facility may use Object Services and other Common Facilities.

2.5.4 Technical Issues

None

2.6 Scripting Facility

2.6.1 Description and Requirements

The Scripting Facility supports:

- A Turing-complete, interpretable language that supports functional decomposition. This is needed to support sending scripts as agents.
- Exposing of key interfaces to key Common Facilities, Object Services and ORB facilities at the language level.
- A visual programming environment (for example, keystrokes recording, mouse clicks recording) to create macros that can be executed by the language.

2.6.2 Related Standards and References

Typical examples of standards are:

Microsoft Visual Basic

Apple Script and Open Scripting Architecture

Scheme, CFI's macro language

TcL, Scripting language popular in the UNIX community

2.6.3 Relationship to Other Components of OMA

Script programs can:

- Be considered as agents in the Task Management Facility.
- Have hooks into the high level messaging service of Task Management Facility.
- Generate high level messages to the messaging service. Use services listed in the advertisement registry of CORBA.
- Send and receive events from the Event Service.
- Create, delete, copy, and move objects using the Life Cycle Service.
- Use the Collection and Relationship Services.

The script visual program environment may use the Application Development facility. In addition, names in the scripting language should be mappable to names in the Naming Service.

The Event, Life Cycle, Relationship, and Naming Services are described in *CORBAservices*.

2.6.4 Technical Issues

Depending on the architecture chosen for scripting a common facility, there may be technical issues. If scripts are considered as agents, then the following issues need to be addressed:

- The set of scripting languages can expand from procedural languages to include declarative languages (rules, logic, and so forth).
- High level messages in task management can themselves be scripts. Therefore, the messaging facility needs to support sending of interpretable programs within it.
- The rules language of task management and the scripting language can be one and the same.
- Management tools for information storage and retrieval need to be able to store and retrieve scripts.
- Information exchange needs to support the exchange of scripts.

Information Management Common Facilities 3

3.1 Overview

The aim of information management is to enable an enterprise to get good value from its investment in information. This includes both codified information held in structured databases and documentary information held in text, image, or some other form. The technology enables information to be modelled, described, stored, retrieved, moved and interchanged within an information system. These functions should be supplied with some quality of service.

Figure 3-1 on page 3-1 shows the topics addressed by the Information Management Facility. The blocks represent the subdivision of the subject area rather than operational relationships between the parts.

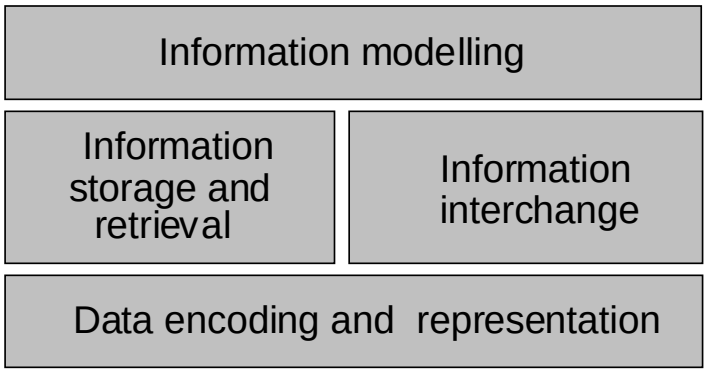

Figure 3-1 Information Management Facilities

Information Modeling

An information modeling facility defines the rules by which information is structured, accessed, and maintained. Associated with an information model is an interface and quality of service. The interfaces reflects the operations defined by various models (such as relational model and object models).

Information Storage and Retrieval

An information storage and retrieval facility embraces all database products, not only business database systems such as Codasyl and relation database managers, but also document storage and retrieval systems for text image, graphics and other media. It includes object oriented databases and interfaces, and directory systems (x.500) used in locating objects, databases, storage system, and so forth.

It also includes repository systems which define and manage the information within the enterprise. Repositories are increasingly dynamic and object oriented. They are used in domains such as data warehousing, application development and systems management.

Information Interchange

Information interchange allows information to be interchanged between different users and different software components. Information needs to converted between different models, media and encodings.

There are several potential common facilities to support information interchange, including:

- A Compound Interchange Facility that provides a framework for the storage and interchange of data objects.
- A Data Interchange Facility that provides basis mechanisms for the interchange of data.
- An Information Exchange Facility.

Data Encoding and Representation

Data encoding and representation standards support practical interworking and interchange of information, via shared storage media, networking protocols or direct programming interfaces. Components wishing to communicate must share not only an information model, but also a common standard for encoding the information with that model.

An important topic with this component is character encoding, which is becoming increasingly important as information systems start to span the globe. Encoding rules are also important for other kinds of information, for example, image, graphics, multimedia documents, and electric data interchange.

The re-encoding or addition of redundant data can be used to change the quality of service. For example, adding redundant data can be used to guarantee the integrity of the data. Re-encoding the data may be used to secure the data from being read by unauthorized users.

One way of supporting data encodings is to encapsulate them. The Time Operations Facility, described on page 3-17 is the first attempt at this approach.

3.1.1 Relationship to Components of OMA

There is a general relationship between information management and all other Common Facilities, and specific relationships with some of them.

The general relationship is that information management provides services to all other Common Facilities, which they use to meet their information needs. For example, data dictionaries and directories should be built with standard information management components.

The specific relationships are described in the following paragraphs.

Vertical Market Facilities contain information management components. These in turn include subcomponents which describe the principal information stores, information models, and information flows, and defining constraints on the way they are to be implemented.

User Interface Facilities should in principle be independent of information management facilities. However, applications are simpler to write if both share the same information model; for example, if they are capable of using the same coordinate system for spatial data. This interaction become very strong when multimedia information is used. ODA, for example, conveys information about the presentation of a document as well as its logical structure. Indeed, for communication between human beings, good presentation is important to ensure that the information is properly understood.

There is some overlap between distributed database technology and **Object Services,** such as distributed transaction processing. The two technologies are to some extent alternative solutions to the same problem.

For example, the Transaction Service has been closely associated with databases technology for many years. The primary interface between the two is the two-phase commit protocol. (The Transaction Service is described in *CORBAservices*.)

Task Management Facilities contain information management components. Task Management needs information architecture describing the principle control flows, control models, rules defining constraints, operational policies and quality of services.

Systems Management Facilities include the management of all components of the information system, including databases. System management of these components includes, operational control, movement, access control, quality of service, measurement (performance and storage capacity) and capacity planning.

3.2 Information Modeling Facility

3.2.1 Description and Requirements

An Information Modeling facility is traditionally defined as the data structure and data manipulation rules used to create a schema. Within the object paradigm, data and operations are modeled together. The Information Modeling Facility in a distributed object system would be used to create part of the type system.

The Information Modeling Facility should be able to describe:

- Object interfaces.
- Services that may be composed of a set of interfaces.
- Relationships between objects.
- Data types required for describing exchanged data and atomic types.

The Information Modeling Facility should feature:

- Graphical editors for drawing various types of diagrams.
- A persistent data repository for organizing and identifying object types.
- A browse or query capability to enable client programs to view the contents of the repository.
- An application program interface to the object repository.
- A type extension capability for new object types.
- A reporting system to support project documentation standards.

The Information Modeling Facility should also address information structuring, access, and integrity.

Information Structuring

Information Structuring should support:

- A uniform modeling approach, applicable through analysis, design and implementation.
- The development of application architectures that include these sources of enterprise information; legacy systems; products of domain analysis; and purchased libraries.
- The OMG Object Model.

Information Access

Information Access should support:

- Security (access control)
- Transaction control
- Referential integrity
- Concurrency control locks
- Multiple views
- Fine grained and course grained objects

Information Integrity

Information Integrity should support:

- Versioning
- Update
- Data dictionaries

3.2.2 Related Standards and References

ANSI X3 X3H7 OO Databases.

ANSI X3 X3H6 CASE Tool Integration Models.

ANSI X3 X3H4 IRDS Repository.

ISO/IEC JTC1 SC7 Software Engineering.

ISO/IEC JTC1 SC21 OSI: Database, SQL, IRDS.

ISO/IEC JTC1 SC21 WG11 Data Definition for Software Engineering.

ECMA TC33 PCTE Portable Common Tools Environment.

ISO/IEC JTC1 SC22 PCTE Portable Common Tools Environment.

ISO/IEC JTC1 SC21 WG7 Open Distributed Processing.

3.2.3 Relationship to Components of OMA

The Information Modeling Facility may depend on:

OMG Object Model
OMG IDL
Relationship Service
Properties Service
Interface Repository
Trader Service
Persistent Object Service
Change Management Service
Transaction Service

The OMG Object Model is described in the *OMA Guide*. The Relationship, Persistent Object, and Transaction Services are described in *CORBAservices;* the Trader and Change Management Services will be described in a future version of *CORBAservices.* OMG IDL and the Interface Repository are described in *CORBA: Common Object Request Broker Architecture and Specification.*

3.2.4 Technical Issues

The Information Modeling System may be used to create a type manager necessary to support a Trader as described in RM-ODP.

3.3 Information Storage and Retrieval Facility

3.3.1 Description and Requirements

The Information Storage and Retrieval facilities comprise the higher level storage and retrieval specifications for distributed applications. These specifications will be applicable to a wide range of information services, including database access and information highways.

Some of the facilities needed to describe a generalized standard interface for information retrieval systems are:

- Initialization service

- Search service
- Retrieve service
- Access-control service
- Termination service

Z39.50 and HTTP are candidates for being an "information interface," but both are limited and might not be able extensible enough to handle all necessary data types and transactions.

Metadata is not only necessary to effectively search, retrieve, and view the information, but it will also enable the sort of access constraints that will be required by various providers for billing, privacy, and security reasons.

Currently, there is no standard for what metadata should be provided or how it should be represented. The Standard Generalized Mark-up Language (SGML) may be a candidate, but it is not suitable for all data types and is not universally adhered to. (The Imagery Generalized Mark-up Language (IGML) MOIE has explored other options for spatial metadata.) HTML+, the standard used by Mosaic and the World Wide Web, is not strictly SGML compliant. There are groups looking at the metadata issues like the Coalition for Networked Information (CNI).

The Repository Facility is a specialization of the Information Storage and Retrieval Facility to provide services to manage information resources including metadata. Typical repository services include:

- Version and configuration management
- Metadata and model management
- Notification of changes
- Constraints and rules management

3.3.2 Related Standards and References

Standard Generalized Markup Language (SGML).

Hyper-Text Markup Language (HTML).

Z39.50 and HTTP.

ANSI/ISO Structured Query Language: SQL89, SQL92, and the draft SQL3.

Open Data Base Connectivity (ODBC).

Wide Area Information Services (WAIS).

World Wide Web (WWW).

3.3.3 Relationship to Components of OMA

Information Storage and Retrieval Facilities are closely related to a number of Object Services:

The **Query Service** will provide a globally API for information retrieval. The IS&R facilities may specialize the Query Service specification to provide higher level services.

The **Transaction Service.** Transactions are a key capability of information storage and retrieval, so the Transaction Service may be used in the specification for Information Storage and Retrieval Common Facility.

The **Persistent Object Service** is the foundation for storage.

The **Security Service** will provide some fundamental specifications for access control and other security issues.

Specifications for the Transaction Service and Persistent Object Service are contained in *CORBAservices*. Specifications for the Query and Security Services will be published in future versions of *CORBAservices*.

3.3.4 Technical Issues

The metadata service must be reconciled with the Interface Repository and the Relationship Service.

3.4 Compound Interchange Facility

3.4.1 Description and Requirements

The Compound Interchange Facility provides a framework for the storage and interchange of data objects to support facilities like the Compound Presentation Facility. The facility roughly maps to the persistent storage subsystem of a compound document architecture.

The Compound Interchange Facility should address:

- Binding of data objects to a particular presentation manager.
- The annotation of these data objects with arbitrary additional information (properties).
- Conversion of data objects to different types.
- Exchange of data objects, both on-line (for example, via drag-and-drop or the clipboard) or off-line (for example, email or via an external media such as a floppy disk).

- A linking facility to pass information from one object to another.
- A reference storage format of these data objects to provide a canonical interchange format.

3.4.2 Related Standards and References

Currently, none.

3.4.3 Relationship to Components of OMA

Currently, none.

3.4.4 Technical Issues

Currently, none.

3.5 Data Interchange Facility

3.5.1 Description and Requirements

The Data Interchange Facility enables objects to interoperate through exchange of data. The Data Interchange Facility is used for many forms and kinds of data transfer, such as:

- Interchange of domain-specific object representations. These object representations might be generated using the Externalization Service. (The Externalization Service is described in *CORBAservices*.)
- Interchange of formatted data, which can include data in file formats, such as TIFF, GIF, EPS, NITF, and so forth, and other formatted data.
- Bulk data transfer.
- Structured data transfer, such as transfer of OMG IDL- specified data types (that is, without externalization).
- Data interchange between encapsulated legacy systems.

In addition, the Data Interchange Facility is also used for data interchange between objects and encapsulated software, for example, interchange between a compound document object and an encapsulated legacy application. In this example, the Data Interchange Facility could provide a more economical mechanism for providing this interoperability, as opposed to complete migration of the legacy system into a compound document object.

Data interchange may be effected, for example, through defining a protocol with which objects publish the representations they support, defining mechanisms for registering representations, and negotiating the best representation for interchanging information. Data Interchange may require first supporting the registration of operations that return important representations, such as any of the numerous graphics formats used for bitmap data or various document or product data interchange formats, and then supporting negotiations that determine which representation is best for the current interchange.

Data Interchange may entail:

Provision for multiple representations. Methodologies for data interchange can vary greatly, depending on the number of representations an object's data may be expressed in, and how dynamically this set of representations may change.

Methods for defining allowable representations. Interface mechanisms supporting multiple representations for data interchange must address (either implicitly or explicitly) how the set of allowable representations is defined. The set may be a static list of just those representations of interest to a particular supplier (or group of suppliers). It may be up to the individual objects to determine which representations they can provide, with the formats determined privately between the requester or supplier, or else as a result of some publishing mechanism.

Publication and registration of allowable representations. Publishing the representations an object supports might be done in a variety of ways. De facto registration might occur via the type system. Alternatively objects might register their supported representations in the trader service or identify them in the implicit invocation context. Objects supporting data interchange may be required to provide an operation that returns specific representations by name, or a set of operations that returns a specific representations. Both the list of allowable representations and/or the publishing mechanism may be based on relevant industry standards such as the ISO Abstract Syntax Notation (ASN.1) or EIA CDIF. These standards may be incorporated into or endorsed for use by data interchange services.

Supporting different qualities of data interchange or different usage paradigms. Negotiation may be necessary to guaranteeing that all information is preserved in an interchange. In the case of bitmap data, for example, resolution or image size and depth may be important factors to data interchange. The intended use of interchanged data, such as display-only representations versus editable representations, may be another important factor in negotiation.

Economy of mechanism. This should be a key design consideration in the Data Interchange Facility. For it to have wide applicability, it must be simple to understand and inexpensive to implement, yet provide substantial interoperability benefits. Implementation should be equally feasible for both commercial software suppliers, corporate end-user developers, and system integrators.

Interfaces supporting formatted data conversions. Since most application objects use unique data formats, conversion of formatted data is essential for interoperability. In most systems conversion software is bundled with application software, so that the conversion

functions are replicated throughout the system. In addition, bundled conversions are inaccessible to other applications, which inhibits interoperability with high fidelity (proprietary) formats. Provision of conversion interfaces allows reuse of conversion functions, which reduces the software cost to provide data interchange, and significantly enhances the potential for interoperability. With conversion services, the Data Interchange Facility can provide outstanding interoperability benefits as a standalone facility.

3.5.2 Related Standards and References

ISO Abstract Syntax Notation (ASN.1)

OLE2 Universal Data Transfer (UDT)

OpenDoc Features related to Data Interchange: Clipboard Objects, Linkage Objects, Drag&Drop Objects

Computer Aided Logistics/Computer Aided Manufacturing (CALS/CAMS)

Electronic Data Interchange (EDI)

EIA CASE Data Interchange Format (CDIF)

File formats for data interchange: RTF, NITF, TIFF, PICT, EPS, Acrobat, etc.

Information Processing and Interchange Functional Specification (ISO 12087-2)

Data Interchange and Synergistic Collateral Usage Study (DISCUS)

3.5.3 Relationship to Components of OMA

The Data Interchange Facility may depend upon:

The **Externalization Service** for object state interchange.

The **Data Interchange Facility** could be used by application developers with the following related services and facilities. It is probably not appropriate for the data interchange service to replicate or specialize these service interfaces, but the Data Interchange Facility should work readily with these services in application systems.

The **Compound Interchange Facility** for data interchange between compound document component parts. Active data interchange such as linking, publish, and subscribe.

The **Change Management Service,** if copies with new representations are considered to be new versions.

The **Relationship Service,** to represent relationships or linkages between exchangeable data. This may involve use of the Event Service to notify of changes to linked data.

The **Data Encodings and Representations Facility**, which may provide services for generating and accessing encoded data.

The **Properties Service** for attaching dynamically defined properties to interchangeable

data.

The **Persistent Object Service** for persistently storing interchangeable data.

3.5.4 Technical Issues

None.

3.6 Information Exchange Facility

3.6.1 Description and Requirements

Information exchange is the ability of information appliances to exchange different kinds of data and operations between themselves. The information exchange can happen from one information appliance to many and from many to one.

Three layers of interchange are considered. These provide the mechanisms for achieving information exchange. However each layer is useful in its own right and may be used without needing to use the higher layers. It is also likely that demands from the highest layer may require some enhancements to the lower layers.

Two different types of information exchange are also considered here: mediated and non-mediated information exchange. Mediated exchange refers to the use of an intermediate program to collect and combine different kinds of data and/or operations to respond to user queries. Non-mediated refers to direct, one on one exchange of data and operations between systems.

Layers of Data and Information Interchange

There are three layers which can be considered as follows:

- Lowest Level "Infrastructure" - ORB Interoperability
- Medium Level "Enabling Technology" - Object Data Interchange Service
- Highest Level "Semantics" - Information Exchange Facility

The lowest level provides the transfer of formatted data. This is a simple way of structuring predefined data which is understood both in terms of format and semantics by both the sending and receiving systems. The structure of the data is defined in OMG IDL and any changes need to be agreed and implemented at both ends.

The medium level consists primarily of the sending of data objects. These provide a means of exchanging a much richer set of information than simple data structures but tend to raise problems of how to identify and understand the methods that should be used to access the data .

A specific case of data objects can provide a Semantic Data Service. In this model standard methods are provided to access and manipulate the data. This enables not just the type of each data item but also its name and context to be sent and inspected. This data structure does not need to be ordered as each item can be identified by its name/context It enables systems to communicate by using a common reference model rather than needing to know specific characteristics of the other party. It does not however try to define the common reference model.

Information exchange is the highest level of interchange between systems. It is not limited to traditional data nor is it limited to the passing of structured sets of parameters. It may result in agents or mediators undertaking searches on behalf of the requester and the results may exist nowhere else in the system.

Mediated Information Exchange

Information exchange is different than object data exchange. Object Data exchange refers to exchanging objects of interest between appliances and operates at the Medium Level defined above. Information exchange, on the other hand, operates at the Highest level and refers to the establishment of a virtual information network between appliances, between appliances and mediators, between mediators themselves. It also refers to the exchange of different types of data including notifications, agent programs, and data through this network.

The virtual information network is established by the use of mediators who assemble a network of information appliances to respond to queries from other appliances or from users. The virtual information network is an active pipe line that constantly maintains the state of the information sources that are connected to it and propagates data and state change information through that network. This virtual information network can be reconfigured at any time.

The types of information that can be exchanged through this virtual information network can include data objects, but is not limited to it. The information can be data objects or agent programs or events or notifications or other types of information not normally considered to be "data" in the context of data exchange.

In the rest of this document, information appliances also refer to mediators unless explicitly differentiated.

Infrastructure

This is assumed to be provided by an Object Request Broker (as defined in *CORBA: Common Object Request Broker Architecture and Specification)* and is not discussed further here.

Object Data Interchange Service

This enables the transmission of semantically enriched data by providing a packaging and communication service between data objects. It provides a general purpose vehicle for

semantic communication. It needs to provide a packet format that is explicitly defined and closely related to the object model used by the Object Data Interchange Service.

The Object Data Interchange Service needs features to support interaction with autonomous systems. Interaction with autonomous systems are by nature asynchronous and connection less. Finally, it needs to support transparent use of underlying security mechanisms.

Information Exchange Common Facility

In order to support mediated information exchange, the following services have to be provided by information appliances or mediators deployed in between them:

- Content language service
- Vocabulary service
- Communication service
- Interaction control service

Content Language Service

In order for two information appliances to be able to talk to each other, they need to support the same language. This is broader than just a file format and schema language. The content language needs to be able to not only represent data instances, but also agent programs and protocol descriptions.

By language we mean a commonly agreed to, computer representable, interpretable, language (preferably with a BNF). This language is used to exchange content and can be different from the language used for communication. The language needs to:

- Support unmbiguous representation of vocabularies.
- Support organization of and mapping between vocabulary libraries.
- Support expression of transaction knowledge between information appliances.
- Use a Vocabulary service, because common vocabulary is needed for information appliances to be able to talk to each other. The vocabulary used is defined in the content language. TheVocabulary service needs to be flexible to support expression of schema, class lattices, and ontilingua. (Ontilingua are organization of data expressed in first order predicate calculus.)

The Vocabulary service needs to support mapping of external concepts into the common vocabulary. External concepts are entities represented in the proprietary representation of the information appliances.

Communication Service

The Communication service is used to package the content language and its vocabularies for transmission between the systems. It uses the Object Data Interchange Service as the general purpose vehicle for this. It needs to support explicit representation of administrative information associated with the communication, the language being used, and the vocabulary being used.

Interaction Control Service

Finally, the information exchange service needs to support establishment of a virtual information network, control the interactions between the information appliances in the network, and manage the network itself. Establishment of a virtual information network is achieved based on the capabilities of the information appliances. The control of interaction involves support for registering of interest in change of information (a publish/subscribe model), control of flow of information between the systems (streaming control), and other types of control. Finally, management of the network involves recruiting new members into the network, cleaning up the network of non participating members, and shutting down the network.

In addition, the interaction control service needs to support several additional functionalities to support interaction between mediators and information appliances. These include multi-casting of messages and facilitation of mediators by mediators.

3.6.2 Related Standards and References

EDI

CCITT Abstract Syntax Notation One (ASN.1)

3.6.3 Relationship to Components of OMA

These relationships have already been described in Description and Requirements on page 3-12.

3.6.4 Technical Issues

None.

3.7 Data Encoding and Representation Facility

3.7.1 Description and Requirements

Data encoding and representation standards can be thought of at the lowest layer of the Information modeling protocol stack. Information storage and retrieval, and information modeling all depend upon the facilities defined for encoding and representing data as it is passed between systems and applications. Data encoding and representation standards

allow for the exchange of information across networks of heterogeneous computer systems by providing a common information model and a common way of encoding information within that model. Encoding must support not only character data, but other sorts of data as well, including imagery, graphics, multimedia documents, and electronic mail.

Many different schemes and algorithms exist for encoding or compressing data, and for representing it in some canonical form as it is passed between heterogeneous systems. Different systems will undoubtedly require different classes of encoding algorithms or representation schemes. For instance, some applications which process imagery may be most concerned with speed, and able to tolerate a very lossy compression algorithm, while another application may be most concerned with undetectable loss compression, and willing to suffer performance penalties. This proliferation of schemes has resulted in the need for a common interface among encoding and representation schemes.

The Data Encoding and Representation Facility should provide interfaces to generalized services for:

- Data compression
- Data decompression
- Conversion from internal representation to canonical representation
- Conversion from canonical representation to internal representation

The interfaces defined should provide a capability to specify the sort of data to be encoded, and quality of service requirements, such as acceptable loss or performance.

3.7.2 Related Standards and References

Multipurpose Internet Mail Standard (MIME) from the Internet Engineering Task Force, RFC 1521 & RFC 1522.

ISO Joint Photographic Experts Group (JPEG) compression standard for imagery.

ISO Motion Picture Experts Group (MPEG) compression standard for video.

Sun Microsystem's eXternal Data Representation (XDR), RFC 1014.

CCITT Abstract Syntax Notation One (ASN.1).

CCITT V.42bis for compression of data transmitted over communications networks.

Huffman Coding algorithm.

3.7.3 Relationship to Components of OMA

The **Trader Service** could be used to select among a range of available compression algorithms and services, and canonical representations.

The **Security Service** may be called upon to guarantee the safe delivery of encoded data.

The **Relationship Service** could be used to indicate a relationship between an original data object and its canonical or compressed form.

The **Change Management Service** might be used to monitor different versions of a data object as it is encoded with different quality of service parameters.

3.7.4 Technical Issues

None.

3.8 Time Operations Facility

3.8.1 Description and Requirements

The Time Operations Facility provides a standard way of representing a Date/Time Group (DTG) as an instance of a time stamp, a duration of time, and a specific window of time between two time stamps (a start and ending time or a start time and a duration). Services should manipulate time instances, perform comparisons and arithmetic operations on time instances. Services should be provided to convert time instances to strings for presentation and printing.

The Time Facility must be able to provide these services:

Set time | date stamp

Set (specify) a duration

Compute duration from a provided time stamp | another duration

Provide time stamp

Comparison of time stamps | durations | time windows (<, =, >)

Determine if a time window falls within another time window (i.e., 2-4 pm within 1-5 pm)

Determine if one time window overlaps another (2-4 pm overlaps 2-3 pm)

Related to a time stamp, provide:

- Day of month
- Days in month
- Days in year
- Day
- Days left in year

- Identify leap year
- Day name
- Month name
- Month index into year (i.e., February = 2)
- Year
- Number of hours | minutes | seconds since midnight

Related to a duration, provide the number of days | hours | minutes | seconds | milliseconds represented.

Set start | end time of time window

Set start time + duration of time window

Compute new time window based upon existing time windows

Related to a time window, provide:

- Start | end time
- Duration

Convert time stamp | duration | window to string for presentation

3.8.2 Related Standards and References

Open Software Foundation, Distributed Computing Environment: Time Service.

3.8.3 Relationship to Components of OMA

This facility is a specialization of the Time Service (see OMG Object Services RFP3).

3.8.4 Technical Issues

None.

System Management Common Facilities 4

4.1 Overview

System Management Common Facilities should provide a set of utility interfaces for system administration functions. These interfaces abstract basic functions such as control, monitoring, security management, configuration, and policy that are needed to perform Systems Management operations, such as adding new users, setting permissions, installing software, and so forth.

The classes of user identified for System Management Common Facilities include:

- Computer System Administrators (User)
- Management Application Developers (Developer)
- System Service Providers (Service Provider)
- Computer System Resource Planners (Enterprise)

System Management Common Facilities have their own unique view of a systems architecture.

Resources are the entities within a system that require management. Resources can include physical entities (such as printers or routers) and logical entities (such as users, groups, applications, and other common facilities).

For a resource to be managed it must provide a management interface or have it provided by a proxy, and take part in management interactions such that it becomes a managed resource.

The OMG has developed an architectural scheme whereby resources within the enterprise are viewed as manageable components. A manageable component may directly invoke a

Systems Management Facility, although usually this class of facility will be used to manage components. Tools will be developed to aid in this management effort.

Systems Management Facilities are as follows:

- Policy Management
- Quality of Service Management
- Instrumentation
- Data Collection
- Security
- Data Collection
- Security
- Collection Management
- Instance Management
- Customization
- Event Management

4.2 Description and Requirements

Policy Management

Policy Management provides a common interface that supports the definition of policy and its application to manageable components. Policy is used to provide control over the creation, deletion, and modification of manageable components.

This facility allows administrators to be able to define their own policy. It also enable administrators to define both the default values for object attributes and the rules to validate the modification of those values.

Policy Management provides the functionality that supports the ability to group resources into policy regions over which a set of policies is defined.

Quality of Service Management

Quality of Service provides a common interface that supports the selection of service levels for availability, performance, reliability and recovery.

Instrumentation

Instrumentation facilities provide a common interface for the gathering, management and dissemination of resource-specific data in support of systems management.

Facilities that might be defined include:

Workload. For example: invocation per second on an object from both application and object point of view; object create counts; object delete counts; clients per object; locates per second.

Allocation of Objects to Physical Resources. For example: object location; object replication; object failure due to host crash, network crash, disk crash;

Responsiveness. For example, invocation response time; invocation throughput; object availability; object system errors; locate delay.

Data Collection

Data Collection facilities provide a common interface to information gathering functionality in support of systems management. Data Collection facilities identified thus far are: Logging and History Management.

Logging provides a common interface that supports the recording and storage of information relevant to systems management. For instance, keeping records of system events.

History Management provides a common interface that supports the storage of and access to archived data.

Any manageable component may have a history, which is a record of historical events related to the component. An example of this kind of facility is an interface that provides for the query of a manageable component's history.

Security

Security Facilities provide a common interface for the management of the security of system resources. This is distinct from the implementation of security mechanisms.

Collection Management

The Collection Management Facility defines operations required to have two way reference relationships between objects. It enables the resulting collections to be queried or have operations applied to the members of the collection. The collection functionality is required to allow administrators to interact naturally with applications.

Instance Management

Instance Management facilities provide much of the required infrastructure for objects to be logically associated and managed with other objects which support a common interface and are subject to common policies. It defines the fundamental operations which support the management of multiple instances of an object type.

Scheduling Management

Scheduling Management requires the performance of certain tasks on a regular basis in a controlled manner. A facility is required to be defined in a flexible and precise manner which reliably performs these tasks under the control of particular stimuli.

Customization

The Customization Facility provides a mechanism for object instances to be extended dynamically while retaining type safety. This allows application objects to be extended and customized once there are instances in place without invalidating existing object references. Examples of this might include replacing a printer with a more functional model.

Event Management

Event Management facilities provide for the generation, registration, filtering, aggregation and forwarding of event notifications to management applications. They must provide facilities for applications to reliably receive notifications using complex filtering rules. Enabling automation of an action associated with events is the key aspect to move to a proactive mode of operation.

Other Services

Other potential System Management Common Facilities include a Process Launch service and a Consistency service. These facilities are under investigation by X/Open and are not yet fully defined.

4.2.1 Related Standards and References

X/Open Resolution 20-6, *Interfaces for a Distributed Systems Management Framework*, (IDSMF).

X/Open Systems Management: Reference Model (XRM).

Tivoli Systems, Tivoli Management Environment (TME), proposal to X/Open SysMan.

Network Management

SNMP. The Internet version of Network management. Over 20,000 MIB variables have been defined.

CMIP. The Telephone Company version of Network Management. OMNIPOINT from Network Management Forum describe the basic Managed Objects Classes.

SNMP to/from CMIP to CORBA Translation:

Joint X/Open NM Forum Inter-domain Management (JIDM) Task Force has produced several documents for converting SNMP (MIBs) and CMIP (GDMO) definitions into OMG IDL.

OIW NMSIG-94/086m Translation of CMIP/ASN.1 Spec to CORBA-IDL, January 24, 1994.

OIW NMSIG-94/085m Translation of SNMPv2 MIB Spec to CORBA-IDL, March 2, 1994.

OIW NMSIG-94/084m Comparison of OSI, OMG, Internet M.O.Models, July 14, 1994.

Middleware Management

OSF DCE OSF (http://www.osf.org:8001/) had a working group on instrumenting DCE. The two main documents are:

R. Friedrich, *Requirements for Performance Instrumentation of DCE RPC and CDS Services*, DCE-RFC-32.0. (http://www.osf.org:8001/dce/dce-rfc/rfc-archive/rfc32.0.txt).

R. *Friedrich, Standardized Performance Instrumentation and Interfaces for the DCE*, DCE-RFC-33.0 (Not published).

System Management

X/Open Systems Management Working Group. This group has a work program that is focused on providing specifications for Distributed Systems Management. It has a published a Reference Model (referenced above), and is currently developing a specification for Systems Management Services necessary to provide a Systems Management Framework based on CORBA. It is also advancing the work of the PMWG (see below) which may soon become an X/Open Preliminary Specification.

Performance Management Working Group (PMWG) of X/Open and Computer Measurement Group (CMG). PMWG is defining a Universal Measurement Architecture (UMA) and has three documents available via anonymous ftp (tarpon.instrumental.com). PMWG is a mature specification and some implementations may appear soon.

Posix Datapool, <pub/pmwg/<pub/pmwg/dcispec4.0.ps>>.

Measurement Layer Interface (MLI),<pub/pmwg/mlispec7.2.ps>.

Data Capture Interface(DCI) <pub/pmwg/dcispec4.0.ps>.

System Management for the PC. Desktop Management Task Force (DMTF) has several documents on how system management works for the PC and with the Desktop Management Interface (DMI). The DMTF mailing list deals with issues related to the Desktop Management Task Force and the Desktop Management Interface. The mailing list is available at this address:

dmtf-info@sun.com

DMTF materials, including a white paper, plan-ahead guide, various press releases, DMI reference code, and draft version 4.5 of the DMI specification, are available via anonymous FTP from:

aurora.intel.com: /pub/dmtf/Spec
gatekeeper.dec.com: /pub/forums/dmtf/Spec

NIST Distributed System Management. In June 1994, NIST has started a new group to work on system management. This effort is focusing on ISO OME. The contact for the work group is:

jhungate@nist.gov.

4.2.2 Relationship to Components of OMA

It is envisioned that hardware vendors will deliver ORBs bundled with their native systems and with a standard set of System Management interfaces (facilities). These facilities would provide the capability to control, configure, and monitor system resources.

Other Common Facilities and Object Services should specify the management interfaces needed in order to manage them. For example, Security and Time Facilities and Services require management. These interfaces should include monitoring, control, and configuration functionality.

4.2.3 Technical Issues

None.

Task Management Common Facilities 5

5.1 Overview

Task Management facilities provide an infrastructure for applications and desktops that model and support the processing of user tasks. The infrastructure contains facilities for managing user and project workflows, rules, and communication. High level messages communicate requests—originated by user transactions in a task-oriented human interface—to Task Management facilities. Figure 5-1 on page 5-1 shows the components of Task Management.

.

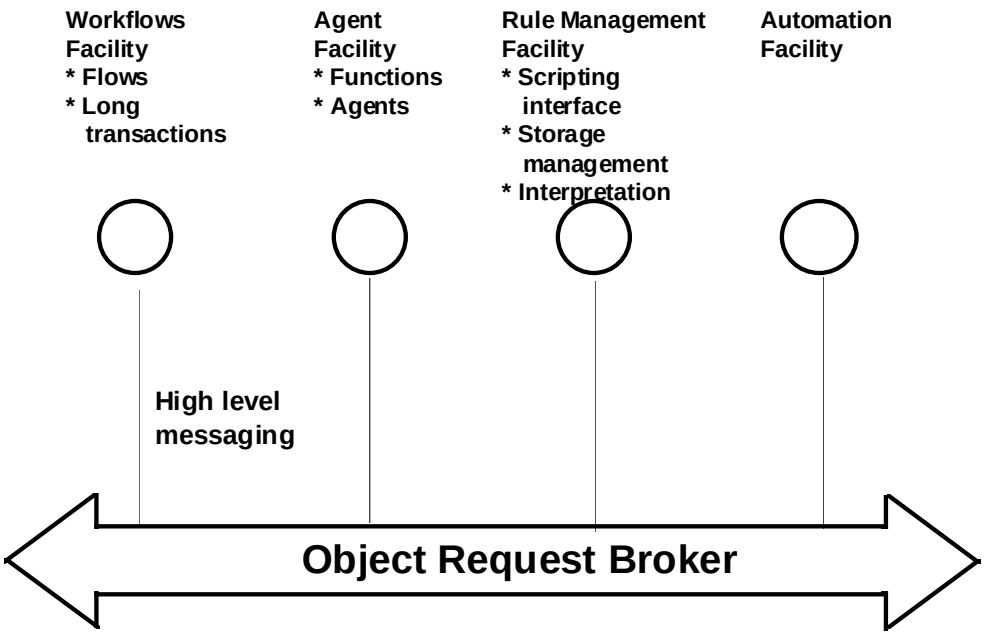

Figure 5-1 Task Management Facilities

Tasks can be composed of a single, short operation such as an ad hoc query or contain many repeatable serial and parallel operations that continue between user sessions, such as group authoring or engineering design projects. Tasks are created, as user or project units of work, by identifying the information object to process and binding this object to a workflow definition. The information object may be a simple structure such as an email message or a complex structure such as a compound document or product design with many levels of hierarchy. Regardless of size and complexity, the information object bound to the workflow represents the object of the task, such as a message, a document, or a product design.

Workflow definitions identify functions that can range from simple tools to intelligent agents. Functions are selected or automatically executed by the evaluation of conditions expressed by rules. The rules specify access (in multi-user workflows), flow sequence, and conditions for executing and terminating functions. Flow sequence, or the marking of steps in a flow, and function initiation/termination are usually determined by evaluation of the bound information object state. Functions are atomic units that contain information specifying how they are invoked within their execution environment. Workflows may be structured in containment hierarchies to reduce complexity and to facilitate inter-operability between heterogenous workflow managers. Workflows may be derived from, and inherit from, other workflows.

With scripting, graphical, and other user interfaces tasks are created and then processed by users interacting with the system. User requests are mapped to a set of high level messages that implement the request. Examples include determine workflow state, query information object state, query access criteria, lock object, and start function. As these messages represent user transactions messaging facilities must assure reliable delivery with store and forward support. Messaging facilities also provide change notification support by event type registration and multi-cast messages. Event notification may be used by Task Management rules to trigger or prompt handling of changes in components of information objects bound to the task.

Task Management, like Information Management, has a general relationship and several specific relationships with other Common Facilities.

The general relationship is that Task Management provides execution services used by all other Common Facilities.

Task Management uses Information Management facilities to create, destroy, relate, access, version, and lock information objects. It is used by User Interface facilities as the infrastructure behind a task-oriented human interface. Task Management Facilities use most of the Object Services, including Event, Life Cycle, Persistent Object, Transaction, Relationship, Query, Concurrency Control, and Change Management. Task Management facilities also support application tasks and system management tasks such as enrolling users and installing code.

5.2 Workflow Facility

5.2.1 Description and Requirements

The Workflow Facility provides management and coordination of objects that are part of a work process for example, purchase process.

The Workflow Facility should provide support for the following:

Production-based workflow, which is a structured, pre-defined process that are governed by policy and procedure. This type of workflow tends to be department or organization specific and is usually implemented by IS teams. Areas in which this type of workflow are applicable are: configuration management, service requests, document routing and review, and "corporate" services such as travel requests, expense vouchers, and purchase orders.

Ad hoc, or coordination-based workflow, which are evolving workflows defined by one or more people to support the coordination of knowledge workers. This type of workflow is not predefined, and is intended to support knowledge workers in their daily activities. Areas in which this type of workflow are applicable are ad hoc tasking and requests; developing a strategic plan; and quick reaction requests.

5.2.2 Related Standards and References

ISO X.400, for mail enabled applications.

Object Data Management Group, for database based group applications.

Workflow Management Coalition, *Glossary*, August 1994.

5.2.3 Relationship to Components of OMA

The Workflow Facility is related to several Object Services. It relies on the:

Relationship Service, to permit the modeling of workflow processes as collection of related objects.

Event Service, to notify and propagate information regarding events that have occurred within a workflow process.

Data Interchange Service, to convert from heterogeneous formats where work processes span users on heterogeneous applications or platforms.

Change Management Service, to maintain audit trails of modifications made by different users to a process or a shared object that is manipulated as part of a process.

Security Service, to limit access to a shared object within a workflow and for invoking certain sub-processes.

5.2.4 Technical Issues

None.

5.3 Agent Facility

5.3.1 Description and Requirements

The agent infrastructure is made up of *static agents* and *mobile agents*. Static agents communicate with each other through mobile agents. Static agents may themselves solve some problem (for example, a mail sorting agent) or may be wrapped around existing applications to provide new functionality (such as a reservation agent that is wrapped around a flight schedule database system). Mobile agents represent the "intelligent message" concept. They are similar to mail messages. As such they may contain data, programs, procedure calls, query scripts, and so forth. In addition, they also contain administrative information to be used by static agents.

Agent infrastructure will address a number of end user and developer problems in the emerging information highway. First, agents off-load bookkeeping and cognitive loads from the end user[1]. Second, static agents can make it possible for existing legacy applications to be "face-lifted" with more capability[2]. This approach not only makes possible integration of legacy systems, but also allows development of large systems in an asynchronous and staged manner. Third, agent infrastructure can support mobile computing[3] and dynamic configuration of operational systems used in military and civilian applications[4].

Agent infrastructure layers higher level semantics and dynamic, behavior based interaction on top of OMG's object based services. In addition, agent infrastructure supports interaction of mobile computing devices with land based computing devices. These functionalities are supported through a logical architecture that modularizes and separates the specification of services for the mobile and static agents.

First, the services are modularized by grouping them as services that mobile agents offer and services that static agents offer. Second, within each of these modular specifications, services are layered.

These services are described in the next section.

Description and Requirements: Mobile Agents

The services of mobile agents are layered to support inter-operability between various types of static agents. First, in order to support integration of legacy applications into an agent network, the mobile agents provide a *content service*. The content service separates the specific content in the mobile agent from other networking and communication information that may accompany it. Applications attached to static agents usually deal only with the content. Second, to support *peer to peer, asynchronous* communication between

1. Schwartz, E.I., *Software Valets That Will Do Your Bidding in Cyberspace,* The New York Times, 9 Jan 94.
2. Wiederhold, G., et. al., *Toward Mega Programming,* Communications of ACM, Nov 1992.
3. Kantrowitz, B. *The Butlers of the Digital Age Will Be Just a Keystroke Away,* Newsweek, 17 Jan 94.
4. Neches, R. *Distributed Collaborative Enterprises*, White Paper, ARPA.

static agents, mobile agents provide communication services. Finally, mobile agents also contain instructions and administrative information through the *message service*.

Content Service

Mobile agents provide a way for sending and receiving contents that are either processed by static agents or (usually) processed by other applications attached to the static agent. The content can be in ASCII or binary format. For example, a content in a mobile agent may be an SQL query (in ASCII) or a bitmap image to be analyzed.

Communication Service

A communication service allows the mobile agent to identify the sender (a static or mobile agent), receiver (another static or mobile agent), and the mobile agent itself. These identification features support true asynchronous, peer to peer communication between static agents.

Message Service

Since static agents may not wish to understand the contents of a mobile agent (in cases where the content is to be dealt with by an application attached to the static agent), the Message service supports a way for the mobile agent to tell the static agent:

- An action to perform (a performative, for example, query, assert, and so forth).
- An optional service to describe the language of the content and the set of words used in that language (its data dictionary).

The static agent will use this information to perform the correct meta level action. For example, a meta level action may be to invoke the services of the application attached to the static agent through CORBA messages. Another may be to update the advertisement registry of CORBA.

Description and Requirements: Static Agents

This section describes static agent services under two categories: basic and extension. Basic services are supported by all static agents. On the other hand, extensions are supported by agents performing various roles and need not be supported by all static agents.

The static agent's role is to process the performative in a mobile agent. The static agent keeps two types of information in its database:

- Its capabilities and services that it supports (information).
- Capabilities and services that have been assigned to it by other agents (goal).

The static agent services are designed to support multiple conductivity, architectural, and communication choices that can be made by application developers. The static agent services support *point to point* conductivity, *multi-casting* conductivity, and *broadcast type* conductivity where the number of receivers is not known. Static agents can be accessed by other agents using explicit addressing (such as the Internet domain addressing scheme),

symbolic addressing (for example, OMG-KQML-AGENT), or using declarative addressing (for instance, the agent most suited to dealing with queries on OMG's publications). Agent services can be implemented using blocking schemes (synchronous) or using non-blocking schemes (non synchronous). Finally, all static agents on a network are assumed to be known at boot-strap time by a static agent or the static agent may support dynamic addition and deletion of static agents on the network.

The basic static agent services are designed to support the following ordered model of interaction between static agents:

- Establish communication context by asserting and discovering the types of interaction that can be fruitfully had between the communicating agents. The basic information services support this need.
- When context is established, query and respond to queries. Static agents may want to control this interaction. Basic and advanced query services, generation services, and assertion services address this step.
- In a complex architecture with multi-casting, point to point communication, static agents may use the services of facilitators, routers, and so forth. Capability definition, networking, and facilitation services support this need.
- Finally, a number of services are provided to support problems that happen in a dynamic architecture. These include database services, error correction services, and so forth.

Basic Services

The services described here, basic services, are expected of all static agents.

Basic Information Services

The static agent supports performatives that can be used to inform other static agents the contents of its information base. The static agents can tell that an information element is present in its information base, can deny , or untell when it denies an earlier tell.

Simple Query Services

A static agent supports the notion of responding to queries with only one answer (instead of a series of them). The server can be used to *evaluate* a query and simplify it; can be asked to *reply* to a query; can be *ask*(ed)-*if* the content in the mobile agent matches any of the items kept in the server's information base; can be *ask*(ed)-*about* all the items that matched the mobile agent's content; return the first set of schema variables of the static agent information base that matched the content of the mobile agent through *ask-one;* return all the schema variables of the static agent information base that matched the content of the mobile agent content; and return non completion status through *sorry.*

Multi-response Query Services

A static agent can return responses as a stream, which when taken together, make up the response. It can *stream-about* all the matches or *stream-all* the schema variables that matched.

Assertion Services

A static agent can be asked to *achieve* (make true) the contents of a mobile agent. Additionally, it can also be asked to unachieve (make un-true) the contents of a mobile agent.

Generation Services

Static agents support performatives to deal with multiple responses. An static agent can ask another to *standby* after collecting the responses to a performative. Additionally, a server can be *ready* to respond to deal with requests for responses to a performative. Further, a server can be asked to send the *next* response or the *rest*. It can also be asked to *discard* responses that have been collected for a performative. Finally, a server can be asked to process a *generator*.

Capability Definition Services

Static agents can advertise their suitability to respond to particular performatives.

Notification Services

Static agents can be used to notify other static agents to perform some function for them. Static agents can subscribe to changes in response to a performative. Additionally, static agents can monitor other static agents.

Extension Services

The Extension Services are supported by some, but not all, static agents.

Networking Services

Static agents can network other static agents. They do this by *register(*ing) themselves and *unregister*(ing) with other static agents or by asking a responding server to *forward* performatives or by *broadcast*(ing) performatives with all the registered servers (multi-casting). Additionally, they can establish a *pipe* so that any performative coming to a static agent will go to the static agent piped to it. Finally, static agents can also *break* a pipe.

Facilitation Services

Static agents can *broker (one and all)* other agents to respond to a performative. Additionally, they can *recommend (one and all)* other agents who are well suited for dealing with a performative and can *recruit (one and all)* other agents to deal with a performative. The static agent will *forward* all the performatives that the recruited agents are suited to processing to the recruited agent(s).

Database Services

Database services are provided to deal with agents that may not understand the notion of truth values. *Insert* adds the content of the mobile agent to the information base of an static agent. *Delete* removes the content of an mobile agent from the information base of a static agent.

Adaptation Services

Static agents can use adaptation services, such as rate or interface, to deal with the different throughput support provided by the underlying transport methods (for example, SLIP vs. 100 MBit ethernet TCP/IP) and to deal with differing frame size requirements of the application (for example, video data versus SQL queries).

Error Correction Services

Static agents can be instructed to adopt different error correction strategies for different types of mobile agents and different networking architectures.

Automatic Re-transmission Services

Static agents can be asked a priori or on demand to re-transmit responses to performatives. Along with the error correction service, this service supports agent communication with mobile receivers over land line and cellular networks.

Registration Service (Home and Visitor)

Static agents will support repositories that can be used by mobile agents to declare their home, current location, and their destination. This information will be used by the static agents to *facilitate* the appropriate mix of resources to address the needs of the mobile agents.

Security Services

Static agents support two different services for security. First, static agents can be asked to encrypt responses. Second, static agents will use the *Registration Service* to control the type of access mobile agents can have to computing resources.

Management Services

Static agents may need to be managed as a computing resource by network administrators. Static agents will provide performatives to be managed by network managers (or software tools used by them).

5.3.2 Related Standards and References

Agent Communication Language Specifications:

Widerfold, G., et al, K*nowledge Query Manipulation Language (KQML) Draft Specification*, (draft) June 1993. ARPA Knowledge Sharing Initiative. Available via ftp from ftp.cs.umbc.edu in pub/kqml/kqml-spec.ps

Background articles:

Geneserth, M. and Ketchpel, S.P, *Software Agents*, Communications of the ACM, July 1994.

Finin, T., et al, *KQML - A Language and Protocol for Knowledge and Information Exchange*, Technical Report CS-94-02, Computer Science Department, University of Maryland, Baltimore. Available via ftp from ftp.cs.umbc.edu at /pub/kqml/kbks.htm and associated .gif files.

5.3.3 Relationship to Components of OMA

The services described in the previous sections were based on an agent communication language and protocol called Knowledge Query Manipulation Language (KQML). In the following sections, we assume KQML to provide detailed responses. However, the answer will also apply to other agent communication language and protocol that support the services described above.

Possible Interactions with Object Services

Event Service

KQML servers (static agents) will be able to use the Event Service (specified in *CORBAservices*.) Since the model of communication assumed in a KQML server is asynchronous, it will be able to use the services provided by the Event Service. However, the KQML server will declare its own specialization of the events and event channels specified in *CORBAservices*. KQML servers need to be implemented as UNtyped *event channel* objects.

Naming and Life Cycle Services

Neither static agents nor mobile agents make any assumptions about naming or life cycle of objects. Therefore, they will be able to use Naming and Life Cycle Services, which are described in *CORBAservices.*

Services to Be Used Without Major Extensions

Archive, Backup/Restore, Instantiation and Activation, Change Management, Implementation Repository, Interface Repository, Licensing, Replication, Security, and Threads.

Services to Be Used With Possible Major Extensions

The Concurrency Control, Externalization, Relationship, and Transaction Services would be used with major extensions. Those services are described in *CORBAservices.* Other services that could be used with extensions include Data Interchange, Data, Operational Control, Properties, Query, and Trader Services.

CORBA

The preferred mode for KQML servers (static agents) to interact with CORBA is through the Dynamic Invocation Interface (DII). KQML messages (mobile agents) can be sent to KQML servers (static agents) through the Dynamic Invocation Interface. The DII is described in *CORBA V2.0*.

OMG Object Model

As previously discussed, CAISS uses the services of the OMG's Object Services and CORBA, according to an object oriented model. However, the exact profiles and components required need further study.

5.3.4 Technical Issues

Static agents present two technical issues: serial protocol and the Transaction and Concurrency Control Services.

Static agents communicate with each other through agents. Contents of mobile agents may contain data, programs or procedure calls (messages). In particular, mobile agents may carry messages that access Object Services. This may imply that Object Services may need to be specified using a serial protocol, rather than, for example, an API.

It must be determined whether the underlying models used in the Concurrency Control and Transaction Services support mobile agent transactions correctly. The concept of concurrency control and integrity constraint management is not well specified for systems that are distributed and autonomous (which static agents are assumed to be). Only a loose notion of concurrency control and integrity constraint management can be supported for the agent interaction model.

5.4 Rule Management Facility

5.4.1 Description and Requirements

The Rule Management Facility provides for declarative event-condition-action rule specification and processing. Rule management involves the acquisition, management and execution of rule.

The Rule Management Facility will provide a rules specification language and rules execution engine.

The Rule Management Facility should address:

- Uses of rules (database management system integrity, constraints, workflow triggers and knowledge base management systems queries).
- Forward chaining, backward chaining, certainty factors.
- Active rules, deductive rules, integrity constraints.
- How rules relate to OMG IDL inheritance hierarchies (for example, right-hand-side of rules as message bodies, rule inheritance).
- Rule maintenance (for example, insertion, deletion semantics).
- External representations and interchange format for rules.
- Federation of rule management (for example, composing rules services, accommodating heterogeneity and distribution in communicating rule services).

5.4.2 Related Standards and References

X3J21 Specification languages Z and VDM.

X3T2 Knowledge Interchange Format (KIF), conceptual graphs and conceptual schema modeling facilities.

ARPA Knowledge Sharing Program including KIF, KQML, Agents, Mediators and Negotiation.

Prolog, OPS5.

KBMS systems like LOOM, Neuron Data.

Active DBMS systems like Postgres, ODE, OSAM, KBMS.

5.4.3 Relationship to Components of OMA

The rule management facility depends upon the following:

- OMG IDL (mandatory).
- For active rule driven systems, the Event Service, as described in *CORBAservices.*
- For distribution, persistence, security, versioning, and meta data repository: the CORBA specification; the Persistent Object Service and the Security Service.

5.4.4 Technical Issues

- Granularity of rule sharing across heterogeneous rule-based systems
- Rulebase validation (for example, detecting cyclic rules, subsumption, inconsistency, redundancy and trade-offs)
- Rulebase evolution
- Interaction of rules and the Transaction Service (the Transaction Service is described in *CORBAservices)*
- Relationship of rules and the Query Service (specifications for the Query Service will be published in a future version of *CORBAservices*)
- Rule scope in organizational hierarchies and active in a temporal interval or a spatial region.

5.5 Automation Facility

5.5.1 Description and Requirements

The Automation Facility is a set of conventions and interfaces that allows access to the key functionality of an object from another object. In one sense this is very similar to the goals of an ORB, but there are two key differences in emphasis for the Automation Facility that require additional conventions and/or facilities.

The design goal for the Automation Facility is to support user visible objects which are larger grained than the typical ORB object. The typical object acted upon by the Automation Facility would be a document, a paragraph, a spreadsheet cell, and so forth. In addition to being "larger," the objects that use the Automation Facility will typically be further separated (in terms of latency) than many ORB objects.

The emphasis of the facility is for objects to expose enough of their capabilities so they may be driven by scripts and macros. In common usage, there are two areas where Automation facilities diverge from normal CORBA ORB method functionality. A "method invocation" should be able to be directly represented as an object, allowing it to be passed to intermediaries, filtered, and modified. In legacy systems this is commonly implemented by having a well defined data structure represent the method, but there is no requirement that this be the case.

The other key element of this facility is the provision of an "object specifier." An object specifier allows the referencing of an object in the server object's context without specifying a concrete object reference. Example object specifiers in a "document" context would be "the third paragraph," or "all green characters." An object specifier should be able to be defined recursively, for example "the third paragraph on the second page."

5.5.2 Related Standards and References

Inside Macintosh—Interapplication Communications, Addison-Wesley Publishing Company, Reading, MA June 1993. ISBN 0-201-62200-9.

5.5.3 Relationship to Components of OMA

Some of the technical items for the Automation Facility could either be implemented as standalone objects or by extending the core CORBA model. There are advantages to each approach. A standalone object is easier to implement and port, while extending CORBA would allow the creation of other services that would benefit from the same functionality. For example, a generic queuing service could be written if an object method invocation could be represented in a standard way; the queuing service could accept all method invocations sent to it and replay them until they succeeded, without having to understand anything about the particular method signature it was actually queuing. Two other services that could be built using this capability are a trace service or an object debugging service.

5.5.4 Technical Issues

None.

Vertical Market Facilities 6

6.1 Overview of Vertical Market Facilities

While the Horizontal Common Facilities address universal needs, the Vertical Market Facilities are standards for interoperability in particular speciality areas, such as vertical markets or industry segments. Each speciality area should represent the needs of an important computing market. The computing market can be defined by particular industry groups, such as OMG Special Interest Groups (SIGs); industry alliances; speciality conferences; standards groups; or other activities. The resulting standards should represent high-value solutions to interoperability problems that would be costly to resolve retroactively.

The Vertical Market Facilities described in this chapter are a sampling of the potential areas to be addressed by the OMG. The list of Vertical Market Facilities is open-ended and will be updated regularly. The OMG encourages new groups to align their plans and specifications towards the OMG standards. In that way, the computing industry can achieve interoperability within and across multiple speciality areas through the OMG standards.

6.1.1 How a Vertical Market Facility is Adopted by the OMG

Any industry segment that wants to propose a technology to the OMG should work toward a consensus within its own group before approaching the OMG with its proposal. In general, Vertical Market Facilities should be processed through the OMG Fast Track, a process that takes 6 months rather than the usual 12.

The OMG Fast Track Process is defined in the Technical Committee Policies and Procedures; these policies are published in the *Object Management Architecture Guide*. (Please note that policies and procedures are slightly revised from time to time; the OMG maintains the most up-to-date policies and procedures.) The Common Facilities Fast Track is initiated by a corporate member of the OMG submitting an unsolicited proposal to the CFTF through the OMG Common Facilities Technology Desk. The proposal must

be accompanied by a letter of intent, signifying the submitters' commitment to commercialize the technology. At a subsequent Common Facilities Task Force (CFTF) meeting, the submitters may be invited to present their proposal for consideration. A formal CFTF vote will determine if the CFTF recommends a Request for Comment (RFC) to be issued by the Technical Committee. The Technical Committee shall then decide if an RFC is to be issued. At the time of RFC issuance, a public comment period begins in which the proposal may be commented upon by OMG members and non-members alike. If significant comment is received, the OMG has the option to withdraw the RFC. The comments are shared with the OMG membership. At an OMG meeting subsequent to the end of the comment period, the OMG TC may vote to recommend the adoption of the technology. If it votes to adopt, then the technology is forwarded to the OMG Business Committee and Board of Directors for final approval. This process is expected to take about six months, whereas the RFP process typically takes 12 to 16 months.

To obtain more information about how to propose a technology as a Vertical Market Facility (or Horizontal Common Facility), contact the OMG at the address provided in the Preface to this book.

6.2 Imagery Facility

6.2.1 Description and Requirements

Imagery facilities will include OMG IDL specifications for access and interchange of imagery and related data. For the purposes of this description, imagery is defined as a two-or- more dimensional array of data that can be derived from sensors or produced artificially. The role of imagery applications is to support the end-user to examine, process, annotate images, and store/display support data in order to create some added value, depending on the application domain.

There are several kinds of applications involved in imagery systems:

- Imagery applications that directly access the image data.
- Imagery archives that store massive quantities of image data.
- Support applications that access imagery related data, which may be of any type including reference imagery, text, formatted documents, database entries, graphics, and so forth.

Applications of all three types must interoperate with each other to provide a comprehensive imagery facility. An appropriate balance of performance concerns and low integration cost must be incorporated in imagery facilities in order to make the OMG standards feasible for adoption by the market, which uses a wide range of commercial and custom-built software, and frequently extend their systems with new component software capabilities. The requirements for Imagery facilities are summarized in the following sections.

General Requirements for Imagery Applications, Imagery Archives

Adopted specifications should support reasonable performance for handling large images. Avoidance of unnecessary copying and movement of data are very important. For example, imagery has massive primary and secondary storage requirements, terabyte sized archives and gigabyte sized high performance local storage are required by many applications. Even simple processing of images may require massive commitments of processing and storage resources.

Imagery applications require support for high resolution I/O devices; the Imagery facilities should support the flexible, low cost interoperability with new devices.

The Imagery facilities should support interchangeability of imagery applications, image archive implementations, and support applications. The encapsulations defined in the Imagery Facilities specification should provide this implementation independence, include the appropriate level of abstraction to mask implementation differences.

New image types must be incorporated and supported as they become available. These new types may require different implementations of processing and retrieval algorithms, as well as, new operations.

Image quality is a key quality of service for image data interchange, particularly relating to compression/decompression algorithms and format conversions that may be transparently applied to image data during communication.

Support must be available for a wide arrange of imagery formats including formal standards, de facto standard and vendor-specific formats. Imagery interchange facilities should support high fidelity conversions between the various formats. Support for reduced resolution data sets and image compression is also required.

Imagery applications relating to spatial data require the use of mensuration services, which could be an important software component interface within the Imagery Facility. Other component software requirements include image understanding algorithms; image perspective transformation; and 3D graphics.

Imagery overlays and graphical annotations should be supported and interchangeable between Imagery Facilities and MAP/GIS facilities.

General Requirements for Support Applications

Support applications may include: special purpose image processing; algorithms; databases; office automation; mapping systems; statistics packages; and so forth. It should be feasible for image applications and support applications to interchange data with high fidelity, minimal integration costs, and reasonable performance. Also, it should be feasible to add custom software to the configuration without excessive costs because the addition of special purpose algorithms is commonplace in the imagery market.

Integration between support applications and imagery applications must be flexible and low cost. New data types from support applications should be attachable to images and specific image coordinates. The specifications should support transmission, storage, retrieval, and parsing of new data types as needed by image applications.

6.2.2 Related Standards and References

ISO/IEC JTC SC24 WG1 Image Processing and Interchange (CD) 12087: Programmers Imaging Kernel; Image Interchange Facility.

DISCUS Technical Report - Reference Technology for OMG Image Facilities.

National Imagery Transmission Format (NITF) - NITF Tech Board (US DOD).

Image Data Formats (numerous) - TIFF, GIF, PBM, RAS, and so forth.

6.2.3 Relationship to Components of OMA

There are many opportunities for reuse of OMG standards in the definition of imagery facilities:

Life Cycle Service. Imagery facilities may define specific Life Cycle operations for imagery objects and support data.

The **Query Service** and **Transaction Service** may be specialized in the Imagery Facility to provide access to imagery archive services.

The **Data Interchange Service** may be specialized in the Imagery Facility to provide data interchange with other services.

The **Relationship Service** may be reused in the Imagery Facility to provide linkages between imagery and support data.

Trader Service. The Imagery Facilities may define specific profiles for definition of Imagery Facilities metadata in the Trader Service. The Trader metadata can define both imagery facilities services and data sources.

Geographic Information Systems (GIS) Common Facility. GIS facilities should be able to operate with Imagery Facilities, and exchange data such as overlays and bitmaps. There is some commonality between GIS and Imagery, but these are distinct markets, with different requirements and these markets commercially support different sets of products.

The Life Cycle, Transaction, Relationship Services are described in *CORBAservices*. Query and Trader Services will be described in a future version of *CORBAservices*.

6.2.4 Technical Issues

A key issue is the handling of large images —perhaps as large as a gigabyte—including moving and copying. There is a potential requirement to provide access to separate high bandwidth communication channels from the normal ORB-supported communication mechanisms to support moving and copying large images across a network. A similar requirement also applies for real-time collaboration between imagery applications end users across a distributed network.

6.3 Information Superhighways Facility

6.3.1 Description and Requirements

The International Information Superhighway (IIS) has been generally defined as a set of electronic networks to include protocols and rules for their use; information repositories connected through these networks; and a collection of tools that provides users and applications transparent access to this information.

CORBA-based common facilities for the IIS will provide a base set of capabilities for the IIS. These facilities will:

- Provide development and integration of services and users with the infrastructure, including a dynamic reconfiguration and tailoring of services.
- Support a commercially-based operation.
- Facilitate investigation of new technologies in the operational environment without affecting operational capability.
- Provide for management of the resources and user base.

IIS facilities that the OMG has identified include commerial operations; resource discovery; intermediaries; teleconferencing; experimentation; and user access.

Commercial Operations

These facilities would provide support for commercial ventures via the IIS. Since many services will eventually be provided on a pay for use service, the ability to support businesses to operate via the IIS is needed. Commerical operation facilities would provide for the following:

Advertisement offers the ability to notify potential customers of available services.

Monitoring provides businesses with mechanisms for conducting market research on the IIS based on monitoring usage of products and services. This could include generation of mailing lists or could facilitate such a capability as another IIS service.

Costing provides the capability to meter usage; calculate costs; establish pricing

schemes; and provide the ability to handle billing and collection.

Resource Discovery

Vast holdings will become available over time with an ever changing "landscape" to be explored. The ability to discover relevant resources may be developed relative to domains of interest, for example, in the spatial and manufacturing areas. The need for common facilities to support such activities is needed. Much work is ongoing on the Internet today to include Gopher, WAIS, World Wide Web, Mosaic, Archie, and so forth. Building on these capabilities and ensuring continued access to their evolution would be supported by resource discovery facilities.

In addition to text and graphics retrieval, there is a need to support the identification of scarce resources for niche communities. Access to resources such as high performance computing and specialized, high volume data sets would be possible with resource discovery facilities, assuming that such assets are available.

Intermediaries

The concept of an intermediary is discussed in the first article listed in References on page 6-7. An intermediary would provide functions needed to interconnect, adapt, and facilitate services offered by others. These intermediaries include the following:

A **broker,** which provides a mechanism for responding to a request from a client, identifying an available service, and acquiring the response. This is at the heart of CORBA. The ability to support this on the scale and complexity of the IIS is yet to be determined. (See Technical Issues on page 6-8.)

An **intelligent agent**, which provides for autonomous access and discovery of resources; performs actions based on discoveries; and performs actions on behalf of one or more interested parties.

A **mediator,** which negotiates services based on a request.

A **trader,** to acquire services on behalf of a client.

Teleconferencing

The ability to exploit telecommunications for various purposes either as a single party or with other parties will be facilitated through some common facilities.

Collaboration provides the ability to establish for varying durations and dynamically changing interested parties collaborative groups that can exchange information in various forms to include text, voice, and video. Should also include an interactive capability that would include groupware, workflow, and whiteboard concepts.

Mentoring provides facilities to help users with the IIS and its services. This could include intelligent tutoring, courseware development and distribution, and on-line training

and support.

Experimentation

The ability to experiment with various configurations and capabilities within the context of an operational IIS will require some common facilities such as temporary configuration and establishment of services; exercising functionality at non-peak times; and migration to permanent configurations.

User Access

Some of the common facilities that should be available for getting access to the IIS include selecting interface level (such as novice or expert); establishing and managing user profiles; associating with various groups; and so forth.

6.3.2 References

Framework for National Information Infrastructure Services, NIST, Gaithersburg, MD 20899, (draft), July 1994.

Putting the Information Infrastructure to Work: A Report of the Information Infrastructure Task Force Committee on Applications and Technology, NIST, Gaithersburg, Md., NIST Special Publication 857, May 1994.

R&D for the NII: Technical Challenges, R&D for the NII: Technical Challenges Proceedings, February 28 - March 1, 1994, Inter-university Communications Council Inc., May 1994.

6.3.3 Relationship to Components of OMA

The Information Superhighways Facility is related to the Security Service (an object service to be described in a future version of *CORBAservices*); a Security Common Facility; a System Management Common Facility; and several System Mangement Common Facilities.

Security Service and Security Common Facility

Various levels of access will be needed to facilitate placement on the IIS while ensuring access by legitimate users. To avoid duplication of such facilities across the IIS, some common security facilities would be needed. These facilities include authentication of users, ensuring information integrity within the IIS context, and maintaining confidentiality of information accessible through the IIS.

System Management Common Facility: Remote Device Management

Availability of the IIS to a broad spectrum of distributed users will facilitate the use of devices from remote locations. (This is already happening today in the health care community). Facilities to establish connections to remote devices, determination of availability, control at various levels of granularity, and the ability to retrieve and analyze data generated are needed. Devices include scientific instruments, virtual reality devices, health monitoring devices that may be in public (for example, a hospital) or private facilities (for example, a home). Access to these devices should be possible in both a push or pull mode.

System Management Common Facility: Service Management

The IIS service industry will be a dynamic, distributed community requiring a collection of common facilities to support them. The following facilities are envisioned:

Establish, to integrate a service into the IIS; this could include establishing grades of services.

Update, to provide the ability to revise a service or provide for multiple versions at different locations.

Dismantle, to remove services from active duty.

System Management Common Facility: Instrumentation

Performing operations across the IIS will require a certain level of sophistication and awareness to avoid paying heavy costs for services invoked and to ensure performance impacts are minimal. Therefore, some preprocessing or instrumentation facilities would be useful. This would include evaluation of time and space for certain requests such as high volume transfers; identification of conditions that could lead to high costs prior to their invocation; determination of ability to initiate desired transfers; identification of multiple ways to handle request with associated cost and quality; and so forth.

6.3.4 Technical Issues

Scale. The IIS will be a massive backbone for distributed, heterogeneous computing. The use of CORBA technology has been untested at the scale envisioned for the IIS. Issues of performance and configurability of services are considered extremely important topics requiring further investigation.

Internet compatibility. A collection of services using a variety of protocols exist on the Internet today. A CORBA-based approach to common facilities should not obviate their use but should demonstrate compatibility and interoperability with these and other evolving efforts.

6.4 Manufacturing Facility

6.4.1 Description and Requirements

Manufacturing (including CIM) represents the integration, through the use of computational resources, of the manufacturing functions and resources within a factory and with other aspects of the business enterprise. These other aspects include such functions as product engineering; process control; quality management; Computer Aided Design (CAD); sales; marketing; finance; environmental safety and health; and labor management.

Manufacturing systems are the computational resources that support this integration capability. Components of manufacturing systems include manufacturing applications; computer and communications hardware; operating system software; databases; and interfaces to manufacturing equipment and other computer systems. These components are typically defined by some architecture that is tailored to solve manufacturing problems in a specific problem domain.

Traditionally, architectures of manufacturing systems were characterized as inflexible, centralized and monolithic, and often proprietary, systems incapable of rapidly responding to changes in process definitions, customer orders or product design. Such systems have impeded making changes quickly in response to a demand.

An effort is underway within the manufacturing community to solve this problem through the design of new systems that respond quickly to change, and are distributed and reusable. A primary requirement of these new systems is that manufacturing applications be able to interoperate, regardless of underlying computing platform.

The Horizontal Common Facilities, as well as the Vertical Market specializations of those facilities, will play a key role in the design of these new manufacturing systems. Common Facilities will provide functionality that can be incorporated across many different application domains, such as manufacturing, design, analysis, simulation, and business practices. These facilities provide support for the vertical integration of the processes within a single enterprise: collaborative concurrent engineering. These facilities also provide support for the inter-organizational integration of manufacturing processes across multiple enterprises: the virtual enterprise. Common Facilities, along with the OMG's Object Services, will support modular, standards-based implementations that promote agility of systems development and modification.

Work has begun in defining Vertical Market facilities for manufacturing. To date, three specific areas of specialization of common facilities have been identified: policy variable management; history management; and product data service.

Policy Variable Management Specializations

Policy Variable Management (a Horizontal Common Facility) supports the configuration and management of systems management variables.

A manufacturing specialization of this facility would support the management of Business Rules. Business Rules are decision rules that describe procedural variations in manufacturing processes. Manufacturing specializations to the Horizontal Common Facility (core facility) would provide for customization flexibility at runtime without requiring modification of application code.

Within each Business Rule exist Policy Variables, which are facility-dependent parameters. Management of Business Rules implies management of Policy Variables as well.

History Management Specializations

History Management (a Horizontal Common Facility) supports the storage and access of archived data.

A specialization of this facility would support the management of event data. Each object in an enterprise may generate notifications of activity, called events. Typically, an event initiates some form of response.

A collection of these events is called a History. It is imperative that key object histories be archived for such purposes as keeping a record of equipment maintenance, performing activity based costing, and so forth. Specializations to the Horizontal Facility would add behavior specific to the requirements of the manufacturing environment, such as real-time access to archived data.

Product Data Service Specialization

Product Data Services support information sharing in a distributed environment that can cross company boundaries. The industrial enterprise has a special dependency on its product model information (product data). For successful intra-organizational integration, product data must be shared throughout a single enterprise. For successful inter-organizational integration, product data must be shared across company boundaries.

The intra-organizational and inter-organizational integration of information technologies depends upon the creation of an overarching standards framework. This framework must look toward the integration and interoperation of standards rather than the development of stand-alone standards and must be international in scope.

Specifically, the Product Data Service must be fully compliant with the STEP standards and implementation methods. It must use the Standard Data Access Interface (SDAI) to keep the service independent of data store technology and to satisfy the product data sharing requirements identified by the SDAI developers. It must also take advantage of the lessons learned in the evolution of the STEP methodology. The STEP Product Data Service must be compliant with a standard for implementation of Distributed Object Technology. The Object Management Architecture, as defined by the OMG in the *OMA Guide*, provides an international consensus standard for this technology.

The STEP Product Data Service will provide:

- Support for concurrent engineering.

- Support for the integration of information technologies in a distributed computing environment that crosses company boundaries.
- An encapsulated object-oriented interface for accessing product data models from distributed systems that crosses company boundaries.
- Fast execution of large models of fine-grained objects.
- An interface that is fully compliant with the STEP standard.

Other Potential Specializations

Candidates for other specializations that could be identified as extensions to existing Horizontal Common Facilities or extensions to Object Services are:

- Non-product related data interchange and conversion in support of manufacturing requirements.
- Support of application interoperability in support of unique manufacturing requirements.
- Synchronization of redundant data in different applications.
- Support of change of processes and procedures.
- Support of distributed processes and procedures, including
  - Remote ("telnet"style) access and execution of applications.
  - Incorporation of remote operations into the manufacturing process flow.
- Support of factory simulation.
- Support of real-time process control. This might include extensions to time services or possibly the OMG Concurrency Control Service in support of real-time requirements.
- Extensions to object rendering, imagery and/or animation within User Interface facilities in support of manufacturing requirements. An example is the requirement that an object "animate" itself. (A machine object would progress through a series of graphical transformations which would correlate to different steps in a process cycle.) This might require interaction with an event service or a Compound Presentation Facility.
- Extensions to document management facilities in support of unique manufacturing, including CIM, requirements such as:

- The notion of recipe. A recipe is a program, rule or specification regarding factory operations and procedures. A recipe is contained and managed within a document. Typically, business rules decide which recipe is to be used for an operation, that recipe is retrieved from the appropriate document and electronically transmitted to a piece of processing equipment.

- Change Control. This is a process of managing document (for example, process specifications) versioning whereby change requests are submitted, reviewed and accepted or rejected. Upon acceptance, the document is versioned and any changes are disseminated throughout the enterprise.

•Extensions to Task Management facilities in support of task and/or object states unique to manufacturing requirements.

•Extensions to Transaction facilities in support of long-lived transactions that correspond to the lifetime of a manufacturing step.

•Extensions to Security facilities in support of the ability to authenticate multiple, serialized users of the same object or facility. For example, "re-login on the fly" by successive plant operators.

•Support of the notion of the "virtual enterprise."

•Support of remote application compliance testing.

•Extensions to future OMG Object Services that will support commercial applications. These extensions will be defined to meet unique manufacturing application needs.

6.4.2 Related Standards and References

SEMATECH, *Computer Integrated Manufacturing Application Framework Specification 1.1*. 93061697D-ENG. Austin, TX.

Standard Data Access Interface Specification -- Part 22. ISO 10303-22, 1994.

6.4.3 Relationship to Components of OMA

CORBA

Any manufacturing (including CIM) specialization to a core Common Facility must not negate that facility's compliance to CORBA.

Common Facilities

By definition, Common Facilities unique to the manufacturing domain should be specializations of existing Common Facilities. If an abstraction does not exist that can be easily specialized, then a new facility should be defined to meet that need.

OMG Object Model

Manufacturing specializations will require objects to exhibit the following characteristics:

- Objects must be given as much intelligence as is practical and assume a proactive role in improving the state of their domain.
- An object in a system should be able to communicate with any other object in that system without knowledge of or regard to its location.

6.4.4 Technical Issues

None.

6.5 Distributed Simulation Facility

6.5.1 Description and Requirements

Facilities for distributed simulation support distributed simulations of air traffic control; war gaming; video games for entertainment; and other diverse market needs for simulation facilities.

The facilities should work together to establish a framework in which such simulations can be built and modified rapidly. These facilities are designed to cooperate with and exploit portions of the Distributed Interactive Simulation (DIS) protocol of the US DoD, which is both a widely supported de facto standard and an emerging international standard developed through the IEEE.

Distributed simulation requires a number of constituent services, as follows:

- Simulation management
- Time management
- Aircraft and vehicle state
- Flight data
- Adaptation

- Environment

Simulation Management

A Simulation Management service supports the following tasks:

- Configuring a simulation: choosing components; specifying their connections; providing adaptation data; allocating components to hosts.
- Instantiating components on their assigned hosts.
- Ensuring that components have been started successfully; have located other services they require; and are ready to begin a simulation.
- Monitoring the health and status of components.
- Commanding an orderly shut-down of all components.
- Assigning a simulation-unique identification to each component.
- Issuing to entity-creating components simulation-unique entity identifications.
- Commanding components to create a checkpoint of their state.

Time Management

Time Management service is not entirely separable from simulation management. We distinguish it for the sake of discussion; however, an implementation may combine these services. Because these facilities support interactive simulation, it is assumed that events visible to users are perceived as occurring in real time. The facility described here is intended apply to components' external interactions and may not apply to components' internal synchronization. The approach envisioned here is that of the DIS standard. Simulation time is understood to include date. Time Management performs these tasks:

- Provides initial simulation time to components before the simulation begins.
- Commands components to start, pause, resume, and stop the simulation clocks.

Aircraft & Vehicle State

The purpose of this service is to provide an object-oriented interface to the entity state services of DIS. It is assumed that DIS entity state PDUs and mechanisms are used to implement this facility. A service to support aircraft and vehicle state performs these tasks:

- Maintains a local database of remote entities, recognizing new entities when they appear, removing entities when they leave or cease being updated. Data to be maintained for each entity is defined by the DIS standard and includes: identity, type, position, velocity, acceleration, attitude, angular velocity, position and velocity of articulated parts.

- Provides a snapshot of the local database on request, having applied appropriate dead reckoning.

- Allows establishment of callbacks for entity creation and deletion.

- Transmits state information for an entity as required by DIS standard.

Flight Data

- Allows components to subscribe or drop subscription to flight plan distribution.

- Publishes initial plans and changes to subscribers.

Adaptation

Adaptation data is the artificial, physical definition of the ATC system: the location of airways, fixes, definition of airspaces, and so forth. These data are static relative to a simulation. An Adaptation service would allow all components to derive adaptation data from a single, consistent source.

Environment

An Environment service provides data about natural phenomena. Some data, such as weather, are dynamic. Other environment data are static, such as terrain data of interest to a radar sensor model.

6.5.2 Related Standards and References

The DIS Vision: A Map to the Future of Distributed Simulation, Version 1, Institute for Simulation and Training, Univ. Central Florida, IST-SP-94-01, May 1994.

IEEE 1278 *Draft Standard for Information Technology--Protocols for Distributed Interactive Simulation Applications, Version 2.0.3*, Institute for Simulation and Training, Univ. Central Florida, IST-CR-93-01, May 28 1993.

6.5.3 Relationship to Components of OMA

These services are Common Facilities, so components adhere to the Common Object Model. Interfaces to these services will be described in OMG IDL, and they will be mediated by an ORB. However, the internals of an implementation of a service may or may not

use an ORB. In particular, the Aircraft & Vehicle State service is intended to use internally the entity state mechanism of DIS.

6.5.4 Technical Issues

None.

6.6 Oil and Gas Industry Exploration and Production Facility

6.6.1 Description and Requirements

The exploration and production (E&P) disciplines of the oil and gas industry takes in the processes of finding and recovering natural resources, usually hydrocarbons. These business processes involve a large quantity of data, complex algorithms, and long-term data storage. The trends in the industry are to unify these processes into a complete life cycle approach, integrating the data and computing resources of several technical disciplines. There are strong and growing traditions of outsourcing field work, purchasing third-party software, and working on projects in multi-company partnerships.

An initiative was begun in 1991—the Petrotechnical Open Software Corporation, Houston and London—by several founding E&P companies to promote standards for the effective sharing of data and the interchangeable use of vendor applications. The goal of this effort is to define a set of software specifications that address the basic computing environment, the management of data, application access to data, data exchange, and user interfaces. The goal of these specifications is to provide E&P organizations improved portability, scalability, and interoperability of E&P technical software.

The OMG's overall architecture for object management in general and the Common Facilities Architecture in particular, provide a foundation for defining such E&P specifications for the distributed object computing world. The application services required for E&P applications can be defined as a combination of industry-specific facilities, specializations of core Common Facilities, and core Common Facilities.

Virtually all of the Common Facilities described in this book are important for E&P applications. There are likely to be opportunities to specialize many of these for E&P use.

6.6.2 Related Standards and References

Petrotechnical Open Software Corporation, S*oftware Integration Platform, Base Computer Standards, Version 2.0,* PTR Prentice Hall, Englewood Cliffs, NJ publication pending.

Petrotechnical Open Software Corporation, *Software Integration Platform, Epicenter Data Model Version 1.0*, PTR Prentice Hall, Englewood Cliffs, NJ 1993.

Petrotechnical Open Software Corporation, *Software Integration Platform, Data Access and Exchange Version 1.0*, PTR Prentice Hall, Englewood Cliffs, NJ 1993.

Petrotechnical Open Software Corporation, *Software Integration Platform, POSC Exchange Format, Version 1.0*, PTR Prentice Hall, Englewood Cliffs, NJ 1993.

Petrotechnical Open Software Corporation, *Software Integration Platform, User Interface Style Guide Version 1.0*, PTR Prentice Hall, Englewood Cliffs, NJ 1993.

6.6.3 Relationship to Components of OMA

CORBA

Any E&P specialization to a Common Facility must not negate that facility's compliance to CORBA.

Common Facilities

By definition, Common Facilities unique to the E&P domain should be specializations of existing core Common Facilities. If an abstraction does not exist that can be easily specialized, then a new core facility should be defined to meet that need.

OMG Object Model

The OMG Object Model will support E&P's requirements for general object behavior.

6.6.4 Technical Issues

None.

6.7 Accounting Facility

6.7.1 Description and Requirements

Computer accounting facilities are used by most business organizations. Accounting is a function involved in all forms of commercial transactions. It involves the exchange of money, the management of payroll, purchases, sales, and online computing charges. Accounting Facilities represent a vertical market area with a very large end-user community.

As in other areas of computing technology, software interfaces in accounting systems are primarily custom-designed and proprietary. No effective standards exist for accounting software inter-operability and the interchange of components. Brittle in-house systems have little cross-platform functioning with other software used in accounting functions, such as commercial office automation software.

The purpose of the Accounting Facility is to remove this complexity from accounting service providers and end users by providing an interoperable, OMA-compliant approach to accounting interfaces across any enterprise.

For the purposes of this description, the Accounting Facility is defined as the means by which charges can be made. For example, for supplying goods or services, offering disk space and on-line time, and giving access to information, from low-level Accounting Facilities right up to the course-grained classical accounting functions which are common to all businesses worldwide (high-level accounting functions). The Accounting Facility must also support flexible storage and retrieval of user-defined accounting data.

Examples of such classical accounting functions are customers, suppliers invoicing, deliveries, and taxation analysis—all of which can leverage Horizontal Common Facilities. These accounting functions exist in virtually all businesses world-wide and are surprisingly consistent between organizations.

Typically, in non-OMA-compliant systems many of these kinds of operations are handled by accounting software, which is usually implemented as a monolithic or classical application program.

9.7.2 Related Standards and References

Generally Accepted Accounting Principles (GAAP).

Generally Accepted Auditing Standards (GAAS).

GAAP and GAAS are published by the Financial Accounting Standards Board (FASB).

Spreadsheet Data Formats: WK1, WK4, WKS, and so forth.

Electronic Data Interchange (EDI) (Standards governing forms transmission for electronic commerce.)

6.7.2 Relationship to Components of OMA

Basic Accounting facilities may incorporate many Object Services including Life Cycle, Trader, Transaction, Security, Query, and Licensing.

In the case of Licensing, the Accounting Facility may provide the high level interfaces that specialize the Licensing Service for the accounting of computing services.

The Accounting Facility is a Vertical Market Facility. However, it is envisaged that much of the functionality will eventually be incorporated from the Horizontal Common Facilities, including Presentation, Management, Information Management, System Management, and Task Management.

The Accounting Facility may also have wide ranging applications in the other Vertical Market Facilities. For example, accounting functions will be needed in the Information Superhighways Facility, and the Distributed Simulation Facility. It is in this way that the

Accounting Facility cuts across most other Vertical Market Facilities within the Common Facilities Architecture.

6.8 Application Development Facility

6.8.1 Description and Requirements

The Application Development Facility covers the selection, development, building and evolution of the applications needed to support an enterprise's information systems strategy.

This reference model is derived from the reference model for Project Support Environments of the Project Support Environment Standards Working Group (PSESWG, pronounced pieces-wig), which is in turn based on the NIST-ECMA reference model of Software Engineering Environments (which was once known as the "toaster model.")

The scope of the reference model is to describe environments that support projects that engineer, develop and manage computer-based systems.

The categories of interfaces and semantics are shown on page 6-19.

Table 6-1

Interface and Semantics Category[1]	Sub-category	Example Tools
Technical engineering	Systems engineering. Software engineering. Process engineering.	A&D CASE tools, CAE. Compiler; debugger; A&D; GUI developer; simulator; prototyper. Modeling, enaction.
Applications components for reuse[2]	Frameworks and patterns. Domain-specific construction.	Taligent, class libraries with behavior.
Technical management	CM, change management, reuse management metrics.	PVS/CVS, MSP interfaces, Oracle for application development.
Project management	Plan, estimate, risk analysis	MS Project
Support	Word processors	FrameMaker, Word
Framework	Infrastructure interfaces	CORBA, COM, PCTE, X/Windows, Tooltalk, BMS, MSP mechanism, CDIF transfer format.

1. Based on PSESWG Reference Model, NIST Special Publication 500-213

2. Addition to NIST published Reference Model

Technical Engineering

Technical Engineering interfaces and semantics support activities related to the specification. design, implementation, and maintenance of systems. In addition to traditional engineering domains, the reference model also considers life-cycle processes to be an area for which an engineering discipline is appropriate, and services related to that domain are

included here as well. The Technical Engineering interfaces are Systems Engineering; Software Engineering; and Life Cycle Engineering.

System Engineering Interfaces

- System Design and Allocation
- System Simulation and Modeling
- System Static Analysis
- System Testing
- System Integration
- System Re-engineering
- Host-Target Connection
- Target Monitoring
- Traceability

Software Engineering Interfaces

- Software Requirements Engineering
- Software Design
- Software Simulation and Modeling
- Software Verification
- Software Generation
- Compilation
- Software Static Analysis
- Debugging
- Software Testing
- Software Build
- Software Reverse Engineering
- Software Re-engineering

- Software Traceabilty

Life Cycle Process Engineering Interfaces

- Process Definition
- Process Library
- Process Exchange
- Process Usage

There are many other engineering domains, such as mechanical, electrical, and manufacturing. Although these are omitted in he present edition of the model, future revisions of the PSESWG reference model may be expanded to include them.

Applications Components for Reuse

This component of the reference model provides object interfaces to support components for reuse.

Framework and Pattern interfaces and semantics include the following:

- Framework components for applications
- Patterns describing framework components
- Class Libraries with default behavior

Domain-Specific Construction interfaces include: domain-specific application construction products.

Technical Management

The interfaces of this area fall into a middle category that partakes of both Technical Engineering and Project Management. These interfaces pertain to activities that are often equally shared by engineers and managers, so they do not clearly fall into one or the other category.

- Configuration Management interfaces
- Change Management interfaces
- Information Management interfaces
- Reuse Management interfaces
- Metrics interfaces

Project Management

The interfaces for Project Management support activities for planning and executing a project. Project planning is the activity by which efforts of all personnel responsible for a project are coordinated and integrated. Coordination and integration occur through a comprehensive plan for fulfilling the project's need in a timely manner and at a reasonable cost. Project planning takes place throughout the life of a project, from the project inception to completion. The steps include assessing customer needs, examination of strategies to meet these needs, and discussion of implications and effects of such strategies, potentially including a plan for producing a proposal.

A project may be carried out by a number of cooperating or subcontracting organizations. If the case, these interfaces support the planning needed to manage the request for and selection of these organizations. After project initiation (such as contract award), detailed planning of project activities may be necessary, together with ongoing monitoring and re-planning of the project to ensure its success.

The interfaces include the following:

- Planning interfaces
- Estimating interfaces
- Risk Analysis interfaces
- Tracking interfaces

Planning a project also requires definition of the processes that will support the project; interfaces related to these activities will also contribute to project management, though in the technical engineering area.

Support

Support interfaces include facilities used by all users. They generally include services associated with processing, formatting, and disseminating human-readable data; they also include services that provide support for use of the computer system itself.

Support interfaces are divided into several categories: Common Support interfaces; Publishing interfaces; User Communication interfaces; and Environment Administration interfaces. (Note that Publishing interfaces are not yet divided into subcategories, as are the other categories of Support interfaces.)

Common Support Interfaces

- Text Processing
- Numeric Processing
- Figure Processing

- Audio and Video Processing
- Calendar and Reminder
- Annotation

User Communication Interfaces

- Mail
- Bulletin Board
- Conferencing

Environment Administration Interfaces

- Object installation and customization
- User and Role Management
- Resource Management
- Status Monitoring
- Diagnostic
- Interchange
- User Access
- Instruction

Framework

Framework interfaces and semantics[1] comprise the infrastructure of an application development environment. They include those interfaces that jointly provide support for applications, CASE tools, etc., and that are commonly referred to as a CASE environment framework. Many of these interfaces are available from platform vendors as extensions to CORBA (for example, early implementations of OMA) and their persistent data facilities but may also be supplied (potentially overlapping in functionality) by other non-OMG-sanctioned infrastructure facilities (such as COM, PCTE, IRDS, platform windowing interfaces, and SQL-3 or ODMG databases). In some cases facilities in these categories are being defined by other Common Facilities (such as System Management).

Framework interfaces include:

1. The source for these descriptions is *Reference Model for Frameworks of Software Engineering Environments*, Edition 3, NIST, 1993.

- Object Management interfaces
- Process Management interfaces
- Communication interfaces
- Operating System interfaces
- User Interface interfaces
- Policy Enforcement interfaces
- Framework Administration interfaces

6.8.2 Relationships with Other Facilities

There are many relationships between components of the reference model for project support environments and other facilities. Framework services has many such infrastructure relationships.

Almost without exception, every element has a two-way relationship with application development. In one direction, application development is used to develop application components which conform to that element of the architecture. In the other, application development must itself exploit the capabilities of that element.

- Application development must be capable of developing applications that conform to a variety of application architectures.
- Application development must itself conform to an application architecture, to enable application developers to use basic office applications such as word processing, drawing and mail.
- **User Interface** facilities, distributed application services, and information management each have associated application development methods. Specific tools support the development of application components that exploit these elements
- **User Interface** facilities provide the means by which the different tools can be integrated through the use of common presentation objects, desktops and process support systems.
- **Information Management** facilities provide services for the manipulation of the information models that application development tools work with (through the dictionary and configuration services).
- The integration and packaging tools must be capable of delivering applications which can be distributed using **System Management** facilities.

6.9 Mapping Facility

6.9.1 Description and Requirements

Maps are becoming a medium of data visualization and interchange. As computer hardware has progressed to the point of allowing storage of large volumes of data as well as the manipulation and visualization of that data, GIS systems are moving from the realm of special-purpose systems to everyday use.

The term “Mapping Services” is used here to refer to those services required for applications which want to access and display geospatial data. Current GIS systems tend to bundle the data access, analysis, and display functions into one monolithic system.

Mapping Services should provide interfaces to data access and display functionality. Analysis is properly left in the realm of modeling and simulation services. lists the other Common Facilities that will handle the analysis portion.

OGIS Ltd. is an organization that has begun the task of specifying means to achieve uniform access to geospatial data as well as to specify mechanisms for managing that data and for constructing analysis systems out of toolbox components which can be supplied from many different vendors. This effort is known as the Open Geodata Interoperability Specification or OGIS.

The Mapping Facility is intended to build on the work of OGIS in order to provide a cohesive means of manipulating the flow of data from the databases through the constructed analysis modules and into the realm of either presentation or dissemination into secondary data products.

Presentation of raw or analyzed mapping data should be independent of where the data came from or how they were analyzed, and further, should be kept independent of the user interface paradigm that is imposed by any one end-user system. In other words, there is no reason that a user of an in-vehicle navigation system should be denied access to the output of a best-route model simply because the in-vehicle system lacks the right set of user interface widgets to run the analysis. The other side of this equation is that the model which produces the route information should be able to pass its output on to any other system component regardless of whether the component is another model, a display, or a database.

The basic requirements of a Mapping Facility are as follows:

- Assist in the formulation of queries into database services within a geographic context.
- Provide access to modeling and analysis facilities and allow subsequent access and display of the resulting outputs.
- Assist in the production of presentations made up of stacked layers representing the results of the queries.

Several common threads run throughout these requirements:

- Awareness of the various data sources available within and outside of a particular organization.
- The ability to access data sources of differing levels of quality, granularity, schema representation, access privileges and many other parameters which affect the interchange of data as well as the ability to differentiate among the price/performance tradeoff issues of accessing this data.
- The ability to "fuse" these data sources into a common spatial framework (that is, map projection, scale, display resolution, and so forth) and to provide a uniform method of presenting this data.
- The ability to tag instances of the data items in such a way as to allow subsequent identification of the source and to allow queries back into the source database for further analysis.

6.9.2 Related Standards and References

There are numerous data standards in the area of GIS and Geospatial data. These are some representative ones:

OGIS Open Geodata Interoperability Specification.

ANSI X3L1 ANSI Committee on GIS.

SDTS (FIPS 173) Spatial Data Transfer Standard.

DIGEST Digital Geographic Exchange Standard.

VPF Vector Product Format (MIL-STD 600006).

SAIF Spatial Archive and Interchange Format.

6.9.3 Relationship to Components of OMA

The Mapping Facility will rely on several of the other Common Facilities. In particular, tight integration is required with any Database Services, Modeling and Analysis Services, and Presentation Services.

6.9.4 Technical Issues

Technical Issues must be discussed in a specific context.

Object Services Specializations *A*

This appendix describes the Internationalization and Security Facilities, which are specializations of two Object Services: the Internationalization and Security Services. In general, these facilities greatly affect the quality of an information system. All Common Facilities should support Internationalization and Security facilities.

The **Internationalization Facility** enables an information system to meet the needs of people who want to use the systems in their own language and want the system to support their cultural conversions for currency, and so forth.

The **Security Facility** enables an information system to withstand threats to security.

A.1 Internationalization Facility

The Internationalization Facility enables users to use an information system or an application in their own language using their own cultural conventions.

From a user's point of view the Internationalization Facility allows an information system to support:

- The user's language of both operating the system and for the information held within the system.
- The user's conventions for handling numeric data including decimal an-***d thousands points.
- The user's conventions for dates and time. For example, some users require a 24 hour clock while other user's prefer a 12 hour clock. Similarly date formats vary in the ways in which day, month year are presented; for example 12 December 1994 or December 12 1994.
- The user's conventions for handling currency. For example: $200 or 200FF.
- The user's conventions for sorting text and comparing text strings.
- The user's conventions for rendering pages of information. For example, Arabic is rendered on the screen in right to left and top to bottom, some Chinese is rendered top to bottom and left to right.

The choice of the language and cultural conventions supported by an application or a system is partly in the hands of the application developer, the service provider and the user.

When the application is installed users will want control over the way in which they operate the application and this in turn may lead to the position where the users actually specify the languages and cultural conventions in which they operate the application.

When service providers install applications, they need to assess the extent to which users will want to vary the languages and cultural conversions associated with an application.

The Internationalization Facility allows application developers to develop internationalised software within an object system.

The following two sections explain how the Internationalization Facility supports both internationalization and localization.

Internationalization

Internationalization (commonly abbreviated “i18n” since there are a total of 20 characters in the word) refers to the process of developing software without the prior knowledge of the language, cultural data or character encoding schemes they are expected to handle. In system terms, it refers to the provision of interfaces that enable internationalised programs to modify their behavior at runtime for specific language operations.

An internationalized object should make no assumptions about the language of character data (text) that is designed to handle. This refers to data generated internally, extracted from or written to persistent storage, and message text used for communication with a user.

In particular, language has implications for the processing of text for such things as character handling and word ordering. This facility should provide interfaces that enable internationalised programs to manipulate text strings according to the language requirements of individual users.

With regard to message text, this facility should enable program messages to be separated from the code, translated into different languages, and accessed by the program at run time.

Cultural data refers to the conventions of a geographic area or territory for such things as date, time and currency formats.

An internationalized program cannot assume these formats in advance and must use facilities provided by the underlying system to determine their setting at runtime.

A character set is a set of alphabetic or other characters used to construct the words and other elementary units of a native language or computer language. A coded character set (codeset) is a set of unambiguous rules that establish a character set and a one-to-one relationship between each character of the set and its bit representation.

For an object to be able to handle text recorded in different coded character sets, it must not make assumptions about the size or bit assignment of character encodings. In particu-

lar it should not be assumed that any part of an area used to store a character code is available for other uses.

Localization

Localization (commonly abbreviated “l10n”) refers to the process of establishing information within a computer system specific to each supported language, cultural data and coded character set combination. Each such combination gives rise to the definition of one locale.

Each locale contains collating sequence information that informs string compare functions about the relative ordering of characters that are defined in the associated coded character set.

Character classification information provides details about the type of character associated with each legal character code; that is, whether it is an alphabetic, upper-case, lower-case, punctuation, control, or space character.

Case conversion refers to information identifying the possible other case of each legal character code.

Language information refers to localization data describing the format and setting of locale specific cultural data.

A message catalog is a file or storage area containing program messages, command prompts, and responses to prompts for a particular language.

An Internationalization Facility provides a way to represent a locale to an object, and to access a string representation for that locale. The facility specifies how character sets, cultural data, language information, character classification, collating sequence, and case conversion will be performed. The facility also specifies how message catalogs are accessed.

A.1.1 Related Standards and References

X/Open Company Ltd., *X/Open Internationalization Guide Version 2*, ISBN 1-85912-002-4, July 1993.

X/Open CAE Specification, *System Interface Definitions, Issue 4*, ISBN 1-972630-46-4, C204, July 1992.

A.1.2 Relationship to Components of OMA

CORBA

The provision of the Internationalization Facility relies on facilities provided by other parts of the architecture. Specific areas of concern are as follows:

- CORBA needs to be extended to handle character strings in the character sets and encodings required to support the many different languages and the interworking between languages.
- CORBA Interface and Implementation Repositories may need to extended so that users can use language conventions, cultural conventions and character sets as criteria for selecting objects.
- The locale identifier could be explicitly passed as another parameter to a called method. However, a reasonable case can be made for adding the locale information to the CORBA method environment parameter.

Object Services

Many of the Object Services will have to be modified to sort the Internationalization Facility. Object Services that may need to changed include:

- Naming
- Persistent Object
- Trader
- Query
- Event
- Time

Other Common Facilities

Internationalization affects most of the Horizontal Common Facilities.

User Interface must render information according the users cultural conventions. It has to be able to handle alternative user languages.

Information Management must handle character sets and collating sequences.

Systems Management has to ensure that the languages and cultural conventions that the user wants to use are supported by the available technology and that there are no inconsistencies.

Task Management must support tasks described using different languages and again needs to coordinate multi-lingual operation across a single task.

A.1.3 Technical Issues

All the aspects of localization discussed here are available on most modern operating systems. There is an issue of where to draw the line between providing a new interface for these facilities, or having the object use the underlying operating system to get at that functionality. Clearly, at least the locale identity needs to be carried across object method invocations. The utility of providing interfaces to the rest of the functionality depends on how CORBA systems will be actually used. A key question is whether the locale identifiers will be unique and unambiguous across the various CORBA platforms encountered during a computation.

A.2 Security Facility

A.2.1 Role in Securing an OMA-Compliant System

Security Common Facilities will be critical elements in securing the expanding array of OMA compliant systems being deployed in both the business and government arena. OMA-compliant security, however, is complex and pervasive, and as such, the requirements and architecture must be considered in totality, not as individual, arbitrary elements. Comprehensive OMA security requires an integral system approach, with Common Facilities as an element of the complete solution. This section describes the role of Security Common Facilities in the overall OMA Security Architecture and provides guidelines for selecting Security Facility candidates.

A.2.2 Requirements

OMA Security Requirements are documented in the *OMG White Paper on Security*, which discusses security issues in a distributed object computing environment and defines the concepts and requirements for securing an OMA-compliant system. Security as a computer technology continues to evolve; therefore, as new security requirements evolve, they will be added to an updated versions of the *OMG White Paper on Security*.

A.2.3 Characteristics

The Security facilities are additional facilities that are particular to a vertical market (Vertical Common Facilities), such as wholesale banking, or a general, cross-market facility (Horizontal Common Facilities) that address higher level application needs for security. Additional functions provided by Security facilities may use core security facilities provided in the ORB or by the Security Object Service, including functions for access control, encryption, audit, and so forth (See Object Services RFP3 submissions for complete list of security services specified). Other Security facilities may also be used. t

In general, Security Common Facilities are distinguished from other security technology that may be supplied under the ORB or Object Services by the following characteristics:

•Technology is required for a particular market segment's requirements; for example, to make business processes secure and to provide building blocks for making such business processes secure.

•Technology provides a Security Facility that can be provided by an application object without affecting the core security of the system; for example:

- Extends core security by adding interfaces more appropriate for a particular need.
- Provides a simplified view of existing, more complex security facilities.
- Builds a higher level security facility by using core security.
- Builds a new security facility for application use that does not compromise the existing security of the system.

• Technology is required to solve other horizontal market problems; for example, heterogeneous system validation and security analysis tools.

A.2.4 Related Standards and References

OMG White Paper on Security.
ISO 7498-2 (CCITT X.800) - OSI Security Architecture.
ECMA TR/46 Security in Open Systems - A Security Framework.
X/Open Distributed Security Framework(draft).
ITSEC - The European Security Evaluation Criteria.
TCSEC - The US Evaluation Criteria.
POSIX P1003.6 (draft) particularly for access control and auditing.
X/Open P308 and S307 and IETF RFC 1508 for GSS-API.
ANSI X9.E9.

A.2.5 Relationship to Components of OMA

Security is pervasive and Security Common Facilities usually will depend on other componenets of the OMA, including the following Object Services:

Life Cycle Service. The Security Common Facilities may define factory interfaces with Lifecycle services for creation and management of security objects.

Trader Service. The Security Facility may define Trader Service profiles for metadata relating to security.

Relationship Service . The Security Common Facility may define specialization of the Relationship Service that implement various forms of secure relationships.

Event Service. The Security Common Facility may use the Event Service.

Licensing Service. The Security Common Facility may be used by the Licensing Service or by Common Facilities to manage end user licensing. Components of Licensing may be use by the Security Common Facility to manage access control.

A.2.6 Technical Issues

The overall Security Architecture (including the Security Facility and the Security Service) will be reviewed over time and may be revised.

Glossary *B*

This section provides a set of terms and definitions used in this document. For additional OMG terms, see the *OMA Guide*, *CORBA*, and *CORBAservices*

abstract Class — A class used only to derive other classes. An abstract class is never instantiated. Compare **concrete class**.

action data — Information stored in the undo object's action history that allows a part to reverse the effects of an undoable action.

action history — The cumulative set of reversible actions available at any one time, maintained by the undo object.

action subhistory — A subset of action data added to the undo object's action history by a part in a modal state. The part can then remove the subhistory from the action history without affecting earlier actions.

action types — Constants that define whether an undoable action is a single-stage action (such as a cut) or part of a two-stage action (such as a drag-move).

activate — To have received the selection focus. A frame activates itself when a mouse-down event occurs within it. On most platforms, a window is activated when it is to the front, or when the cursor passes over it.

active frame — The frame that has the selection focus. Editing takes place in the active frame; it displays the selection or insertion point. The active frame almost always has the keystroke focus.

active part — The part displayed in the active frame. The active part controls the part-specific palettes and menus, and its content contains the selection or insertion point. The active part can be displayed in one or more frames, only one of which is the active frame.

active shape — A shape that describes the portion of a facet within which a part expects to receive user events. If, for example, an embedded part's used shape and active shape are identical, the containing part both draws and accepts events in the unused areas within the embedded part's frame.

ancestor — See **superclass.**

application — A software product that allows a user to create and manipulate documents. See also **application component, conventional application.**

application component — An OpenDoc component that the user employs to create, edit, or view document parts. Part editors and part viewers are application components; they replace the functionality of **conventional applications.**

arbitrator — An OpenDoc object that manages negotiation among parts about ownership of **shared resources.** Examples of such resources are the menu focus, the selection focus, the keystroke focus, and the serial ports.

auxiliary storage unit — Compare main storage unit.

audience — The kind of consumer (caller) of the interface. An interface may be intended for use by the ultimate user of the service (functional interface), by a system management function within the system (system management interface), or by other participating services in order to construct the facility from disparate objects (construction interface).

base class — See **superclass.**

bearer The kind of object that presents an interface. an object may fundamentally be characterized by the fact that it has a given interface (a specific object bears an interface), or an object may have an interface that is ancillary to its primary purpose in order to provide vertain other capabilities (a general object bears the interface).

Bento A document storage architecture, built on top of a platform's native file system, that allows for the creation, storage, and retrieval of compound documents. The OpenDoc storage system on some platforms is based on Bento.

bias transform A transform that is applied to measurements in a part's coordinate system to change them into **platform-normal coordinates.**

binding The process of selecting an executable code module based on type information.

border See **frame border.**

bundled frame A frame whose contents do not respond to user events. A mouse click within a bundled frame selects the frame, but does not activate the frame.

canvas The platform-specific drawing environment on which frames are laid out. Each window or printing device has one drawing canvas. See also **static canvas** and **dynamic canvas.**

category See **part category.**

change ID (1) A number used to identify a particular instance of Clipboard contents. (2) A number used to identify a particular instance of link source data.

child class See **subclass.**

circular link A configuration of links in which changes to a link's destination directly or indirectly affect its source.

class — A programming entity comprising data structures and methods, from which objects that are instances of the class are created.

class hierarchy — The structure by which classes are related through **inheritance.**

Clipboard — A system-maintained buffer that provides a facility for transferring data within and across documents.

Clipboard focus — Access to the Clipboard. The part with the Clipboard focus can read from and write to the Clipboard.

clip shape — A shape that defines the limits of drawing within a facet.

clone — To copy an object and all its referenced objects. When you clone an object, that object plus all other objects to which there is a **strong persistent reference** in the cloned object are copied.

close — For a frame, to remove it from memory but not from storage. A closed frame is not permanently removed from its document. Compare **remove.**

Common Object Request Broker Architecture (CORBA) — A standard promulgated by the Object Management Group industry consortium for defining interactions among objects.

component — A software product that functions in the OpenDoc environment. Part editors, part viewers, and services are examples of components. See also **application component, service component.**

compound document — A single document containing multiple heterogeneous data types, each presented and edited by its own software. A compound document is made up of **parts.**

concrete class — A class designed to be instantiated. Compare **abstract class.**

construction interfaces

Define the operations that are used to communicate be3tween the core of a Common Facility and related objects that must particupate in providing the facility. they are typically defined by the facility and inherited an dimplemented by participants in the facility. Objects that participate in a facility must support these interfaces.

container

(1) A holder of persistent data (documents), part of an OpenDoc **container suite.** (2) See **container part, container application.**

container application

An application program that has been modified to support embedding of OpenDoc parts. A container application functions as both document shell and part editor for the root part.

container part

A part that is capable of embedding other parts or links to other parts. Compare **noncontainer part.**

container suite

A set of OpenDoc classes that implement persistent storage. The container suite consists of containers, documents, drafts, and storage units.

containing frame

The display frame of a containing part. Each embedded frame has one containing frame; each containing frame can have one or more embedded frames.

containing part

The part in which a frame is embedded. Each embedded frame has one containing part; each containing part has one or more embedded frames.

containment

A relationship between objects wherein an object of one class contains a reference to an object of another class. Compare **inheritance.**

content	See **part content.**
content area	The potentially visible area of a part as viewed in a frame or window. If the content area is greater than the area of the frame or window, only a portion of the part can be viewed at a time.
content element	A user-visible data item presented by a part's content model. Content elements can be manipulated through the graphical or scripting interface to a part.
content extent	The vertical dimension of the content area of a part in a frame. Content extent is used to calculate **bias transforms.**
content model	The specification of a part's contents (the data types of its content elements) and its content operations (the actions that can be performed on it and the interactions among its content elements).
content object	A content element that can be represented as an object and thus accessed and manipulated through semantic events.
content operation	A user action that manipulates a content element.
content storage unit	The main storage unit of the Clipboard, drag-and-drop object, link source object, or link object.
content transform	The composite transform that converts from a part's content coordinates to its canvas coordinates.
conventional application	An application that directly handles events and opens documents, and is wholly responsible for manipulating, storing, and retrieving all of the data in its documents. Compare **application component.**
coordinate bias	The difference between a given coordinate system and **platform-normal coordinates.** Coordinate bias typically involves both a change in axis polarity and an offset.
current draft	A specially designated draft that is the most recent draft of a document.

current frame — During drawing, the frame that is being drawn or within which editing is occurring.

customizable — Characteristic of a scriptable part that also defines content objects and operations for interface elements such as menus and buttons.

derived class — See **subclass.**

descendant — See **subclass.**

destination content — The content at the destination of a link. It is a copy of the **source content.**

destination part — A part that displays, through a link, information that resides in another part. Compare **source part.**

dispatcher — The OpenDoc object that directs user events and semantic events to the correct part.

dispatch module — An OpenDoc object used by the dispatcher to dispatch events of a certain type to part editors.

display frame — A frame in which a part is displayed. A part's display frames are created by and embedded in its containing part. Compare **embedded frame.**

document — In OpenDoc, a user-organized collection of parts, all stored together.

document part — See **part**.

document process — A thread of execution that runs the document shell program. The document process provides the interface between the operating system and part editors: it accepts events from the operating system, provides the address space into which parts are read, and provides access to the window system and other features.

document shell — A program that provides an environment for all the parts in a document. The shell maintains the major document global databases: storage, window state, arbitrator, and dispatcher. This code also provides basic document behavior like document creation, open, save, print, and close. OpenDoc provides a default document shell for each platform.

document window — A window that displays an OpenDoc document. The edges of the content area of the window represent the frame border of the document's root part. The OpenDoc document shell manages opening and closing document windows. Compare **part window.**

draft — A configuration of a document, defined at a certain point in time by the user. A document is made up of a set of drafts.

drag and drop — A facility of OpenDoc that allows users to apply direct manipulation to move or copy data.

drag-copy — A drag-and-drop operation in which the dragged data remains at the source, and a copy is inserted at the destination.

drag-move — A drag-and-drop operation in which the dragged data is deleted from the source and inserted at the destination.

drawing canvas — See **canvas.**

dynamic canvas — A drawing canvas that can potentially be changed, such as a window that can be scrolled or paged to display different portions of a part's data. Compare **static canvas.**

edit-in-place — See **in-place editing.**

editor — See **part editor.**

editor of last resort — The part editor that displays any part for which there is no available part editor on the system.The editor of last resort typically displays a gray rectangle representing the part's frame.

editor preferences — A dialog box, accessed through the Edit menu, in which the user can view and change preferences for the part editor of the currently active part.

embed To place one part within another so that, although its data is stored with the containing part's data and its frame is contained within the containing part's frame, it retains its identity as a separate part. Compare **incorporate.**

embedded content Content displayed in an embedded frame. A part editor does not directly manipulate embedded content. Compare **intrinsic content.**

embedded frame A frame that is part of a containing part's content, and within which an embedded part is displayed. The embedded frame itself is considered intrinsic content of the containing part; the part displayed within the frame is considered embedded content of the containing part.

embedded-frames list A containing part's list of all the frames embedded within it.

embedded part A part that is embedded in another part. The data for an embedded part is stored within the same draft as its containing part. An embedded part is copied during a duplication of its containing part. An embedded part may itself be a containing part, unless it is a **noncontainer part.** Compare **linked part.**

exception An execution error or abnormal condition detected by the runtime facilities of the system.

exclusive focus A focus that can be owned by only one frame at a time. The selection focus, for example, is exclusive; the user can edit within only one frame at a time. Compare **non-exclusive focus.**

externalize See **write.**

external transform A transform that is applied to a facet to position, scale, or otherwise transform the facet and the image drawn within it. The external transform locates the facet in the coordinate space of its frame's containing part. Compare **internal transform.**

extracted draft A draft that is extracted from a document into a new document.

facet An object that describes where a frame is displayed on a canvas.

factory method A method in one class that creates an instance of another class.

fidelity	The faithfulness of translation attained (or attainable) between data of different part kinds. For a given part kind, other part kinds are ranked in fidelity by the level at which their editors can translate its data without loss.
focus	A designation of a shared resource such as menus, selection, keystrokes, and serial ports. The part that owns a focus has use of that shared resource.
focus set	A group of foci requested as a unit.
frame	A bounded portion of the content area of a part, defining the location of an embedded part.The edge of a frame marks the boundary between intrinsic content and embedded content.A frame can be a rectangle or any other, even irregular, shape.
frame border	A visual indication of the boundary of a frame.The appearance of the frame border indicates the state of the frame (active, inactive, or selected).The frame border is drawn and manipulated by the containing part or by OpenDoc, not by the part within the frame.
frame coordinate space	The coordinate space in which a part's frame shape, used shape, active shape, and clip shape are defined.
frame group	A set of its display frames that a part designates as related, for purposes such as flowing content from one frame to another. Each frame group has its own **group ID;** frames within a frame group have a **frame sequence.**
frame negotiation	The process of adjusting the size and shape of an embedded frame. Embedded parts can request changes to their frames, but the containing parts control the changes that are granted.
frame sequence	The order of frames in a frame group.
frame shape	A shape that defines a frame and its border, expressed in terms of the frame's local coordinate space.
frame transform	The composite transform that converts from a part's frame coordinates to its canvas coordinates.

functional interfaces Define the operations invoked by users of a Common Facility. The audience for these interfaces is the facility consumer, the user of the facility. These interfaces present the functionality (the useful operations) of the facility.

generic object An object, relative to some given Common Facility (or Object SErvice), whose primary purpose for existing is unrelated to the Common Facility whose interface it carries. The notion is that the Common Facility is provided by having, in principle, any type of object inherit that Common Facility interface and provide an implementation of that interface. This extends the definition that appears in the *OMA Guide* to include Common Facilities.

graphics system A specific drawing architecture. Some graphics systems (such as Display PostScript) are available on more than one platform; some platforms support more than one graphics system).

group ID A number that identifies a frame group, assigned by the group's containing part.

icon A small, type-specific picture that represents a part. Possible iconic view types for displaying a part include as a (standard) icon, small icon, or thumbnail; the other possible view type is in a **frame.**

identity transform A transform that has no effect on points to which it is applied.

implementation repository Supports the management of object implementations. It provides operations used to install, manage, retireve, and describe the implementations of type interfaces.

inclusions list A list of part kinds that can be embedded in a given part. A part can define and use its own inclusions list in order to restrict embedding into itself.

incorporate To merge the data from one part into the contents of another part so that the merged data retains no separate identity as a part. Compare **embed.**

inheritance A relationship between classes wherein one class (the subclass) shares the type and methods of another class (the superclass).

in-place editing Manipulation by a user of data in an embedded part without leaving the context of the document in which the part is displayed—without, for example, opening a new window for the part.

inside-out activation — A mode of user interaction in which a mouse click anywhere in a document activates the smallest possible enclosing frame and performs the appropriate selection action on the content element at the click location. OpenDoc uses inside-out selection. Compare **outside-in activation.**

instance — See **object.**

instantiate — To cause an object of a class to be created in memory at runtime.

interface repository — Supports runtime access to OMG IDL-specified definitions such as object interfaces and type definitions. It supports adding, locating, searching, retrieving, and managing definitions; type checking object invocation; and dynamic construction of object invocations.

Interface Definition Language (IDL) — A syntax created by IBM to describe the interface of classes that can be compiled by the SOM compiler.

internalize — See **read.**

internal transform — A transform that positions, scales, or otherwise transforms the image of a part drawn within a frame. Compare **external transform.**

interoperability — Access to an OpenDoc part or document from different platforms or with different software systems.

intrinsic content — The content elements native to a particular part, as opposed to the content of parts embedded within it. Compare **embedded content.**

invalidate — To mark an area of a canvas (or facet, or frame) as in need of redrawing.

invalid shape — The area of a frame, facet, or canvas that needs redrawing. Update events cause redrawing of the invalid area.

invariant — An aspect of the internal state of an object that must be maintained for the object to behave properly according to its design.

ISO string — A null-terminated 7-bit ASCII string.

keystroke focus — The destination of keystroke events. The part whose frame has the keystroke focus handles keystroke events. See also **selection focus.**

keystroke focus frame	The frame to which keystroke events are to be sent.
kind	See **part kind.**
layout	The process of arranging frames and content elements in a document for drawing.
link	(1) A persistent reference to a part or to a set of content elements of a part. (2) An OpenDoc object that represents a link destination.
link destination	The portion of a part's content area that represents the destination of a link.
link source	The portion of a part's content area that represents the source of a link.
link specification	An object, placed on the Clipboard or in a drag-and-drop object, from which the source part (the part that placed the data) can construct a link if necessary.
link status	The link-related state (in a link source, in a link destination, or not in a link) of a frame.
lock	To acquire exclusive access to. A part must lock a link source object or link object before accessing its data.
main storage unit	The storage unit that holds the contents property (prop_contents) of a part. A part's main storage unit, plus possibly other storage units referenced from it, holds all of a part's content.
management interfaces	Used for communication between managed objects and System Management Common Facilities (or Object Services). They handle services such as operational control, installation, and deployment.
member function	See **method.**
method	An operation that manipulates the data of a particular class of objects.

modal focus	The right to display modal dialog boxes. A part displaying a modal dialog must first obtain the modal focus, so that other parts cannot do the same until the first part is finished.
monitor	A special use of a dispatch module, in which it is installed in order to be notified of events, but does not dispatch them.
monolithic application	See **conventional application.**
name space	An object consisting of a set of text strings used to identify kinds of objects or classes of behavior, for code-binding purposes. For example, OpenDoc uses name spaces to identify part kinds and categories, and to identify object extensions.
name-space manager	An OpenDoc object that creates and deletes **name spaces.**
noncontainer part	A part that cannot itself contain embedded parts. Compare **container part.**
nonexclusive focus	A focus that can be owned by more than one frame at a time. OpenDoc supports the use of nonexclusive foci. Compare **exclusive focus.**
object	A programming entity, existing in memory at run time, that is an individual specimen of a particular **class.**
object specifier	A designation of a content object within a part, used to determine the target of a semantic event. Object specifiers can be names (“blue rectangle”) or logical designations (“word 1 of line 2 of embedded frame 3”).
OLE 2.0	Object Linking and Embedding, Microsoft Corporation’s compound document architecture.
outside-in activation	A mode of user interaction in which a mouse click anywhere in a document activates the largest possible enclosing frame that is not already active. Compare **inside-out activation.**
overlaid frame	An embedded frame that floats above the content (including other embedded frames) of its containing part, and thus need not engage in frame negotiation with the containing part.

override To replace a method belonging to a superclass with a method of the same name in a subclass, in order to modify its behavior.

owner For a canvas, the part that created the canvas and attached it to a facet. The owner is responsible for transferring the results of drawing on the canvas to its parent canvas.

parent canvas The canvas closest above a canvas in the facet hierarchy. If there is a single off screen canvas attached to an embedded facet in a window, for example, the window canvas (attached to the root facet) is the parent of the off screen canvas.

parent class See **superclass.**

part A portion of a compound document; it consists of document content, plus—at runtime—a part editor that manipulates that content. The content is data of a given structure or type, such as text, graphics, or video; the code is a part editor. In programming terms, a part is an object, an instantiation of a subclass of the class Part. To a user, a part is a single set of information displayed and manipulated in one or more frames or windows. Same as **document part.**

part category A general classification of the format of data handled by a part editor. Categories are broad classes of data format, meaningful to end-users, such as “text”, “graphics” or “table”. Compare **part kind**.

part container See **container part.**

part content The portion of a part that describes its data. in programming terms, the part content is represented by the instance variables of the part object; it is the state of the part, and is the portion of it that is stored persistently. To the user, there is no distinction between part and part content; the user considers both the part content alone, and the content plus its part editor, as a part. See also **intrinsic content, embedded content.** Compare **part editor; part.**

part editor An application component that can display and change the data of a part. It is the executable code that provides the behavior for the part. Compare **part content, part viewer.**

part ID An identifier that uniquely names a part within the context of a document. This ID represents a storage unit ID within a particular draft of a document.

part info (1) Part-specific data, of any type or size, used by a part editor to identify what should be displayed in a particular frame or facet and how it should be displayed. (2) User-visible information about a given part, displayed in a dialog box accessed through a menu command.

part kind A specific classification of the format of data handled by a part editor. A kind specifies the specific data format handled by, and possibly native to, a part editor. Kinds are meaningful to end-users, and have designations such as such as “MacWrite 2.0” or “QuickTime 1.0”. Compare **part category**.

part property One of a set of user-accessible characteristics of a part or its frame. The user can modify some part properties, such as the name of a part; the user cannot modify some other part properties, such as part category. Each part property is stored as a distinct **property** in the storage unit of the part or its frame.

part table A list of all the parts contained within a document, plus associated data.

part viewer A part editor that can display and print, but not change, the data of a part. Compare **part editor.**

part window A window that displays an embedded part by itself, for easier viewing or editing. Any part that is embedded in another part can be opened up into its own part window. The part window is separate from, and has a slightly different appearance than, the **document window** displaying the entire document the part is embedded within.

persistence The quality of an entity such as a part, link, or object, that allows it to span separate document launches and transport to different computers. For example, a part written to persistent storage is typically written to a hard disk.

persistent reference A number, stored somewhere within a storage unit, that refers to another storage unit in the same document. Persistent references permit complex runtime object relationships to be stored externally, and later reconstructed.

platform A hardware/software operating environment. For example, OpenDoc is being implemented on the Macintosh, Windows, and OS/2 platforms.

platform-normal coordinates — The native coordinate system for a particular platform. OpenDoc performs all layout and drawing in platform-normal coordinates; to convert from another coordinate system to platform-normal coordinates requires application of a **bias transform.**

position code — A parameter (to a storage unit's Focus method) with which you specify the desired property or value to access.

presentation — A particular style of display for a part—for example, outline or expanded for text, or wire-frame or solid for graphic objects. A part can have multiple presentations, each with its own rendering, layout, and user-interface behavior. See also **view type.**

promise — A specification of data to be transferred at a future time. If a data transfer involves a very large amount of data, the source part can choose to put out a promise instead of actually writing the data to a storage unit.

property — In the OpenDoc storage system, a component of a storage unit. A property defines a kind of information (such as "name" or "contents") and contains one or more data streams, called **values,** that consist of information of that kind. Properties in a stored part are accessible without the assistance of a part editor. See also **part property.**

protocol — The programming interface through which a specific task or set of related tasks is performed. The drag-and-drop protocol, for example, is the set of calls that a part editor makes (and responds to) in order to support the dragging of items into or out of its content.

proxy content — data, associated with a single embedded frame written to the Clipboard (or drag-and-drop object or link-source object), that the frame's original containing part wanted associated with the frame, such as a drop shadow or other visual adornment. Proxy content is absent if intrinsic content as well as an embedded frame was written.

purge — To free non-critical memory, usually by writing or releasing cached data. In low-memory situations, OpenDoc can ask a part editor or other objects to purge memory.

read — For a part or other OpenDoc object, to transform its persistent form in a storage unit into an appropriate in-memory representation, which can be a representation of the complete object or only a subset of it, depending on the current display requirements for the object. Same as **internalize;** compare **write.**

reference	A pointer to (or other representation of) an object, used to gain access to the object when needed.
reference count	The number of references to an object. Objects that are reference-counted, such as windows and parts, cannot be deleted from memory unless their reference counts are zero.
release	To delete a reference to an object. For a reference-counted object, releasing it decrements its reference count.
remove	For a frame, to permanently delete it from its document, as well as from memory. Compare **close.**
revert	To return a draft to the state it had just after its last save.
root facet	The facet that displays the root frame in a document window.
root frame	The frame in which the root part of a document is displayed. The root frame shape is the same as the content area of the document window.
root part	The part that forms the base of a document and establishes its basic editing, embedding, and printing behavior. A document has only one root part, which can contain content elements and perhaps other, embedded parts. Any part can be a root part.
root window	See **document window.**
save	To write all the data of all parts of a document (draft) to persistent storage.
select	To designate as the locus of subsequent editing operations. If the user selects an embedded part, that part's frame border takes on an appearance that designates it as selected. The embedded part itself is not activated at this stage.
selection focus	The location of editing activity. The part whose frame has the selection focus is the active part, and has the selection or insertion point. See also **keystroke focus.**
service or **part service**	An OpenDoc component that provides services to document parts, instead of creating or viewing them. Compare **part editor.**

shape A description of a geometric area of a drawing canvas**.**

shared resource A facility used by multiple parts. Examples of shared resources are the menu focus, selection focus, keystroke focus, and serial ports. See also **arbitrator.**

sibling A frame or facet at the same level of embedding as another frame or facet within the same containing frame or facet. Sibling frames and facets are **z-ordered** to allow for overlapping.

source content The content at the source of a link. It is copied into the link and thence into the **destination content.**

source frame (1) An embedded frame whose part that has been opened up into its own **part window.** (2) The frame to which other **synchronized frames** are attached.

source part A part that contains information that is displayed in another part through a link. Compare **destination part.**

specific object An object, relative to some Common Facility (or Object Service), whose purpose is to provide a part of the Common Facility whose interface it carries. The notion is that ia limited number of implementations (and perhaps instances) of these objects exist in a system, commonly known as “servers.”

split-frame view A display technique for windows or frames, in which two or more facets of a frame display different scrolled portions of a part’s content.

static canvas A drawing canvas that cannot be changed once it has been rendered, such as a printer page. Compare **dynamic canvas.**

stationery A part that opens by copying itself and opening the copy into a window, leaving the original stationery part unchanged.

storage system The OpenDoc mechanism for providing persistent storage for documents and parts. The storage system object must provide unique identifiers for parts as well as cross-document links. It stores parts as a set of standard properties plus type-specific content data.

storage unit	In the OpenDoc storage system, an object that represents the basic unit of persistent storage. Each storage unit has a list of properties, and each property contains one or more data streams called values.
storage-unit cursor	A prefocused storage unit/property/value designation, created to allow swift focusing on frequently accessed data.
storage unit ID	A unique identifier of a storage unit within a draft.
strong persistent reference	A persistent reference that, when the storage unit containing the reference is cloned, causes the referenced storage unit to be copied also. Compare **weak persistent reference**.
subclass	A class derived from another class (its **superclass**), from which it inherits type and behavior. Also called derived class or descendant.
subframe	A frame that is both an embedded frame in, and a display frame of, a part. A part can create an embedded frame, make it a subframe of its own display frame, and then display itself in that subframe.
subsystem	A broad subdivision of the interface and capabilities of OpenDoc, divided along shared-library boundaries. The OpenDoc subsystems include shell, storage, layout, imaging, user events, and semantic events. Individual OpenDoc subsystems are replaceable.
superclass	A class from which another class (its **subclass**) is derived. Also called ancestor, base class, or parent class. See also **inheritance.**
synchronized frames	Separate frames that display the same representation of the same part, and should therefore be updated together. In general, if an embedded part has two or more editable display frames of the same presentation, those frames (and all their embedded frames) should be synchronized.
thumbnail	A large (64-by-64 pixels) icon used to represent a part. The icon is typically a miniature representation of the layout of the part content.
token	A short, codified representation of a string. The session object creates tokens for ISO strings.

transform — A geometric transformation that can be applied to a graphic object when it is rendered, such as moving, scaling, or rotation. Different platforms and different graphics systems have transforms with different capabilities.

translation — The conversion of one type of data to another type of data. Specifically, the conversion of data of one part kind to data of another part kind. The translation object is an OpenDoc wrapper for platform-specific translation capabilities. Note that translation can involve loss of **fidelity.**

translator — A software utility, independent of OpenDoc, that converts data from one format to another. A translator may, for example, convert text in the format used by one word processor into a format readable by a different one. The translation capability of OpenDoc relies on the availability of translators.

undo — To rescind a command, negating its results. The Undo object is an object that holds command history information in order to support the Undo capability of OpenDoc.

used shape — A shape that describes the portion of a frame that a part actually uses for drawing; that is, the part of the frame that the containing part should not draw over.

user event — A message, sent to a part by the dispatcher, that pertains only to the state of the part's graphical user interface, not directly to its contents. User events include mouse clicks and keystrokes, and they deliver information about, among other things, window locations and scroll bar positions.

user-interface part — A part without content elements, representing a unit of a document's user interface. Buttons and dialog boxes, for example, can be user-interface parts.

validate — To mark a portion of a canvas (or facet, or frame) as no longer in need of redrawing. Compare **invalidate.**

value — In the OpenDoc storage system, a data stream associated with a property in a storage unit. Each property has a set of values, and there can be only one value of a given data type for each property.

viewer — See **part viewer.**

view type	The basic visual representation of a part. Supported view types include frame, icon, small icon, and thumbnail.
weak persistent reference	A persistent reference that, when the storage unit containing the reference is cloned, is ignored; the referenced storage unit is not copied. Compare strong persistent reference.
window	An area of a computer display in which information is presented to users in a graphic user interface, typically containing one or more content areas and controls, such as scroll bars, enabling the user to manipulate the display. Window systems are platform-specific.
window canvas	The canvas attached to the root facet of a window. Every window has a window canvas.
window-content transform	The composite transform that converts from a part's content coordinates to its window coordinates.
window-frame transform	The composite transform that converts from a part's frame coordinates to its window coordinates.
window state	An object that lists the set of windows that are open at a given time. Part editors can alter the window state, and the window state can be persistently stored.
write	For a part or other OpenDoc object, to transform its in-memory representation into a persistent form in a storage unit. Same as **externalize;** compare **read.**
z-ordering	The front-to-back ordering of sibling frames used to determine clipping and event handling when frames overlap.

References for CORBAfacilities

The following documents were used as reference material for *CORBAfacilities.* For more information about OMG specifications, contact the OMG at the addresses listed in the Preface of this book.

Cargill, C. F. *Information Technology Standardization: Theory, Process, and Organizations,* Digital Press. 1989.

Mowbray, T.J. and Zahavi, R. *The Essential CORBA: Systems Integration Using Distributed Objects.* John J. Wiley & Sons, New York, NY, 1995.

Object Management Architecture Guide. Revision 3.0, John J. Wiley & Sons, Inc., New York, NY, June 1995.

CORBA: *Common Object Request Broker Architecture and Specification*. Version 2.0, Object Management Group, July 1995.

Common Facilities Roadmap, Object Management Group, OMG TC Document 95-1-32.

CORBAservices: Common Object Services Specification. Revised Edition, Object Management Group, March 31, 1995.

OMG Common Facilities Request For Information. OMG TC Document 94.2.11, February 2, 1994.

Submissions to OMG Common Facilities Request For Information. Responses received from Apple Computer, IBM, ICL, IMA, MITRE, and Objectory, June 1994.