Open Blueprint

Open Blueprint Technical Overview

GC23-3808-02

Open Blueprint

Open Blueprint Technical Overview

GC23-3808-02

Note!

Before using this information, be sure to read the general information under “Notices” on page vii.

Third Edition (Dec 1996)

This edition applies to the Open Blueprint until otherwise indicated in new editions or Technical Newsletters.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address given below.

IBM welcomes your comments. A form for reader's comments appears at the back of this publication. If the form has been removed, address your comments to:

International Business Machines Corporation
Department CGMD,
P. O. Box 12195
Research Triangle Park, North Carolina
27709-2195
United States of America

FAX: (International Access Code)+1+800+227+5088
Internet: VSIB2HPD at IBM.VNET.COM

IBMLink (United States customers only): CIBHORCF at RALVM13
IBM Mail Exchange: VSIB2HPD at IBMMAIL

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Figures

Notices

References in this publication to IBM* products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Subject to IBM's intelluctual property or other legally protected rights, any functionally equivalent product, program, or service may be used instead of the IBM product, program, or service. Evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, is the user's responsibility.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
500 Columbus Avenue
Thornwood, NY 10594
USA

Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries:

9037 Sysplex Timer
Advanced Function Presentation
AFP
AIX
AnyNet
APPN
AS/400
CICS
Distributed Relational Database Architecture
DRDA
ES/9000
IBM
IMS
IPDS
Language Environment
MO:DCA
Open Blueprint
OS/2
Presentation Manager
RACF
System Object Model
VisualAge
Workplace Shell

The following terms are trademarks of other companies:

Adobe	Adobe Systems, Incorporated
Apple (name and logo)	Apple Computer, Incorporated
ATM	Adobe Systems, Incorporated
AT&T	American Telephone & Telegraph Company, Incorporated
CompuServe	CompuServe Incorporated and H&R Block, Incorporated
C++	American Telephone & Telegraph Company, Incorporated
CORBA	Object Management Group, Incorporated
DCE	The Open Software Foundation
Digital	Digital Equipment Corporation
Encina	Transarc Corporation
Gopher	University of Minnesota
Hewlett-Packard	Hewlett-Packard Company
HP	Hewlett-Packard Company
IPX	Novell, Inc.
Java	Sun Microsystems, Inc.
Kerberos	Massachusetts Institute of Technology
Lotus Notes	Lotus Development Corporation
Macintosh	Apple Computer, Incorporated
Motif	Open Software Foundation, Inc.
Netscape (logo)	Netscape Communications Corporation
NetWare	Novell, Inc.
Network File System	Sun Microsystems, Inc.
NFS	Sun Microsystems, Inc.
Novell	Novell, Inc.
OpenDoc	Component Integration Laboratories
OpenGL	Silicon Graphics, Incorporated
Open Software Foundation	Open Software Foundation, Inc.
OSF	Open Software Foundation, Inc.
POSIX	Institute of Electrical and Electronic Engineers
PostScript	Adobe Systems Incorporated
Solaris	Sun Microsystems, Inc.
Sun	Sun Microsystems, Inc.
Sun Microsystems	Sun Microsystems, Inc.
Taligent	Taligent, Incorporated
Tivoli	Tivoli Systems, Inc., an IBM Company
TME 10	Tivoli Systems, Inc., an IBM Company
Transarc	Transarc Corporation
UniForum	/usr/group
UNIX	X/Open Company Limited
Windows NT	Microsoft Corporation
XTI	X/Open Company Limited
X/Open	X/Open Company Limited
X Windows	Massachusetts Institute of Technology
X Windows System	Massachusetts Institute of Technology

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc., in the U.S. and other countries.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

Microsoft, Windows, and the Windows 95 logo are trademarks or registered trademarks of Microsoft Corporation.

About This Book

This book provides a technical overview of the Open Blueprint structure. The Open Blueprint structure describes a distributed systems environment and provides the base upon which to build, execute, and manage distributed applications. Parts of this book are similar to the *Introduction to the Open Blueprint: A Guide to Distributed Computing*, G326-0395, which provides a higher level description of the Open Blueprint structure.

What's New in this Edition

This edition describes an enhanced version of the IBM Open Blueprint structure.

- Network computing support enhancements:
  - Web Browser resource manager
  - HTTP resource manager
  - Virtual Machine resource manager supporting Java
  - Enhanced TCP/IP function
  - Expansion of security functions to include public key support
  - Expansion of Directory resource manager to include LDAP support and multiple federated directory services
  - Internet voice telephony function
  - Network access to the functions of Application/Workgroup and Data Access Services resource managers
- Collaboration resource manager support includes both synchronous and asynchronous collaboration
- Intelligent Agent resource manager
- Internationalization resource manager
- Systems Management support based on the Tivoli TME 10 structure
- Mobile computing considerations are included
- Multimedia support is restructured
- User Interface resource manager renamed Human Computer Interaction resource manager to reflect industry terminology and enhanced function
- References to OSI transport services have been removed
- URLs are provided for referencing standards

Who Should Use This Book

This book is written for system designers, architects, strategists, and implementers in the information systems industry and in other industries that use information systems. The reader should already be familiar with the concepts of system architecture.

Overview of Contents

This book is organized as follows:

- Chapter 1, “Overview” introduces the Open Blueprint structure
- Chapter 2, “Distributed System Characteristics” describes a broad set of distributed system characteristics and objectives
- Chapter 3, “Open Blueprint Concepts” describes the major concepts of the Open Blueprint structure, such as resource managers, platforms, and standards
- Chapter 4, “Resource Managers” describes the Open Blueprint resource managers
- The Appendix summarizes the standards and protocols specified in the Open Blueprint structure
- A bibliography is also included

Additional Information

The component description papers in the *Open Blueprint: Technical Reference Library* provide more detailed information about each Open Blueprint component. See “Bibliography” on page 87 for ordering information.

Chapter 1. Overview

IBM's Open Blueprint structure addresses the challenges of the open environment by viewing a system[1] as part of a distributed network and viewing the network as if it were a single system.

The Open Blueprint structure serves four major roles:

- It helps users think about, discuss, and organize products and applications in an open, distributed environment.
- It describes IBM's directions for products and solutions in the open, distributed environment.
- It guides developers as they meet users' needs by supplying products and solutions that include functions written to appropriate standards and that can be integrated and can interoperate with other installed products.
- It provides a context for the incorporation of new technologies into a distributed environment.

A goal of the Open Blueprint is to provide consistency among IBM and related products such that they work together to achieve a high level of systemic value. Since the wants and needs of users include openness and product/vendor heterogeneity, the Open Blueprint is based on a combination of existing and emerging industry standards. The fundamentals that allow this support are heterogeneous network support, distributed security and directory function, participation in systems management as per the defined set of standard interfaces and protocols; and, for object-oriented implementations, adherence to the Object Management Group Common Object Request Broker Architecture and Common Object Services as defined by OMG CORBA 1.1 and 2.0.

IBM believes that other vendors will want to provide equivalent levels of integration and interoperability based on the same fundamentals and standards identified in this book, and that the IBM and non-IBM implementations will provide a broad base of function to support applications and additional services.

The Open Blueprint structure enables the network of operating systems to function as a unit, as a **distributed operating system**. A distributed operating system is comprised of multiple systems that are separated from each other and are connected by a communication network. The distributed operating system supports generic distributed computing which is a superset of both client/server and public (Internet) and private network computing. In the distributed operating system enabled by the Open Blueprint structure, each individual system logically contains the facilities described in this book. Based on the subset of the facilities actually contained, a system can be configured as a client or a server system. For more information on configuration possibilities, see “Platforms” on page 20.

Just as an operating system provides the management of resources on a single system, a distributed operating system provides for the management across the network of the same types of resources: files, databases, printers, transactions, software packages, documents, and jobs. This book describes how the equivalent facilities in each system work together to provide support for distributed client/server and network computing applications. Figure 1 on page 2 represents one instance of a system in the distributed operating system. Figure 2 represents a distributed operating system where each individual system is structured according to the Open Blueprint.

---

1 When the word **system** is used by itself, without any other qualification, it means a single hardware platform and its supporting software. The single system can be a cluster of computers or a parallel computer acting as if it were one system.

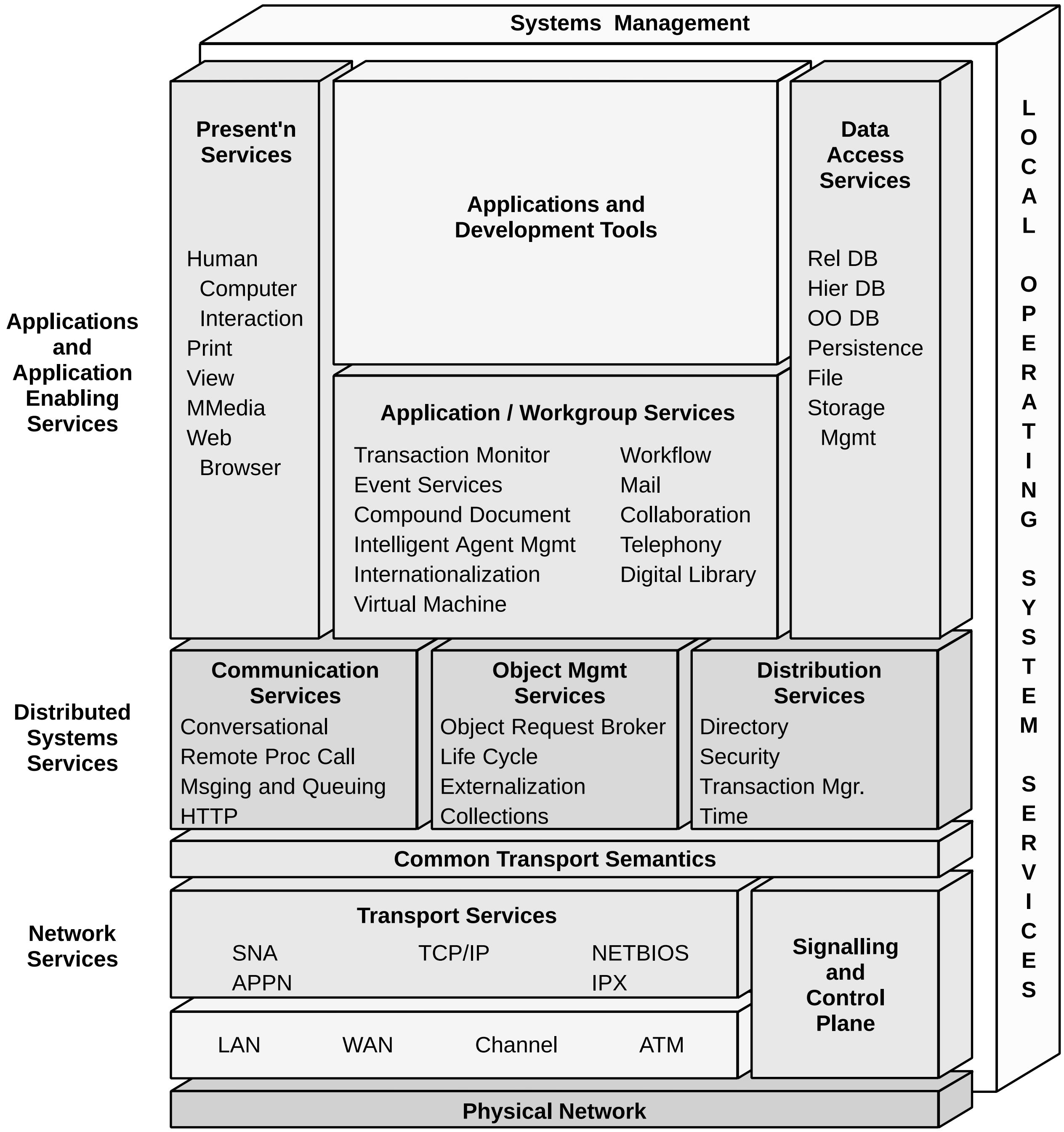

Figure 1. The Open Blueprint Structure

The sets of resource management services in the Open Blueprint are:

Network Services

- **Common Transport Semantics** supports protocol independent communication in the distributed network.
- **Transport Services** provides the protocols for transporting information from one system to another, such as TCP/IP, SNA/APPN, NetBIOS, and IPX.
- **Signalling and Control Plane** provides the ability to establish subnetwork specific connections.

- **Subnetworking** provides functions dealing with specific transmission facilities, such as various kinds of LANs, WANs, channels, Asynchronous Transfer Mode (ATM), and emerging technologies, such as wireless.

Distributed System Services

- **Communication Services** provide mechanisms for parts of a distributed application or resource manager to talk to each other.
- **Object Management Services** provide common object services including transparent access to local and remote objects.
- **Distribution Services** assist the communication between parts of distributed applications and resource managers by providing common functions, such as a directory and security.

Application Enabling Services

- **Presentation Services** define the mechanisms for interaction between applications and users, including Web browser, GUI, and multimedia support.
- **Application/Workgroup Services** are common functions, such as collaboration, which are available for use by all applications.
- **Data Access Services** enable applications and resource managers to interact with various types of data.

Systems Management provides facilities that enable either system administrators or automated procedures to manage the distributed operating system.

Local Operating System Services operate within a single system in a network. Examples of local services are lock management and dispatching work.

Development Tools help the application developer implement distributed applications that use standard interfaces and the facilities of the Open Blueprint.

The Open Blueprint structure promotes the integration of multivendor systems and simplifies the more cumbersome aspects of distributed computing. Products that align with the Open Blueprint provide designated interfaces and protocols. Products that use resource managers defined within the Open Blueprint, rather than their own unique mechanisms, are truly integrated with the Open Blueprint structure. For example, if every application uses its own unique security facility, a user must present a unique password for each application. If applications and resource managers use the security services from the Open Blueprint, a single user password will suffice for the cooperating applications. This integration improves the single-system image that the end user and application developer perceive for the distributed system.

The Open Blueprint structure includes both procedural and object-oriented interfaces and standards from both IBM and other industry sources. Those from industry sources are broadly accepted standards. Many of these standards are the same ones included in X/Open's Architectural Framework (XAF) and OMG's Common Object Services specification.

The resource manager names in the Open Blueprint diagram (as shown in Figure 1 on page 2) do not correspond to specific products. The Open Blueprint structure is implemented by different products on different system platforms. Also, the Open Blueprint does not describe how the implementing software is packaged into offerings. Rather, the Open Blueprint describes technical attributes and characteristics of supporting software, reflects desirable functional modularity, provides software principles and guidelines, and specifies important boundaries and interfaces.

Much of the function described in the Open Blueprint exists and is being developed and used in product form. Over time, the Open Blueprint will be expanded with additional function, and additional product implementations will be provided. This book is conceptual in nature and provides technical direction only.

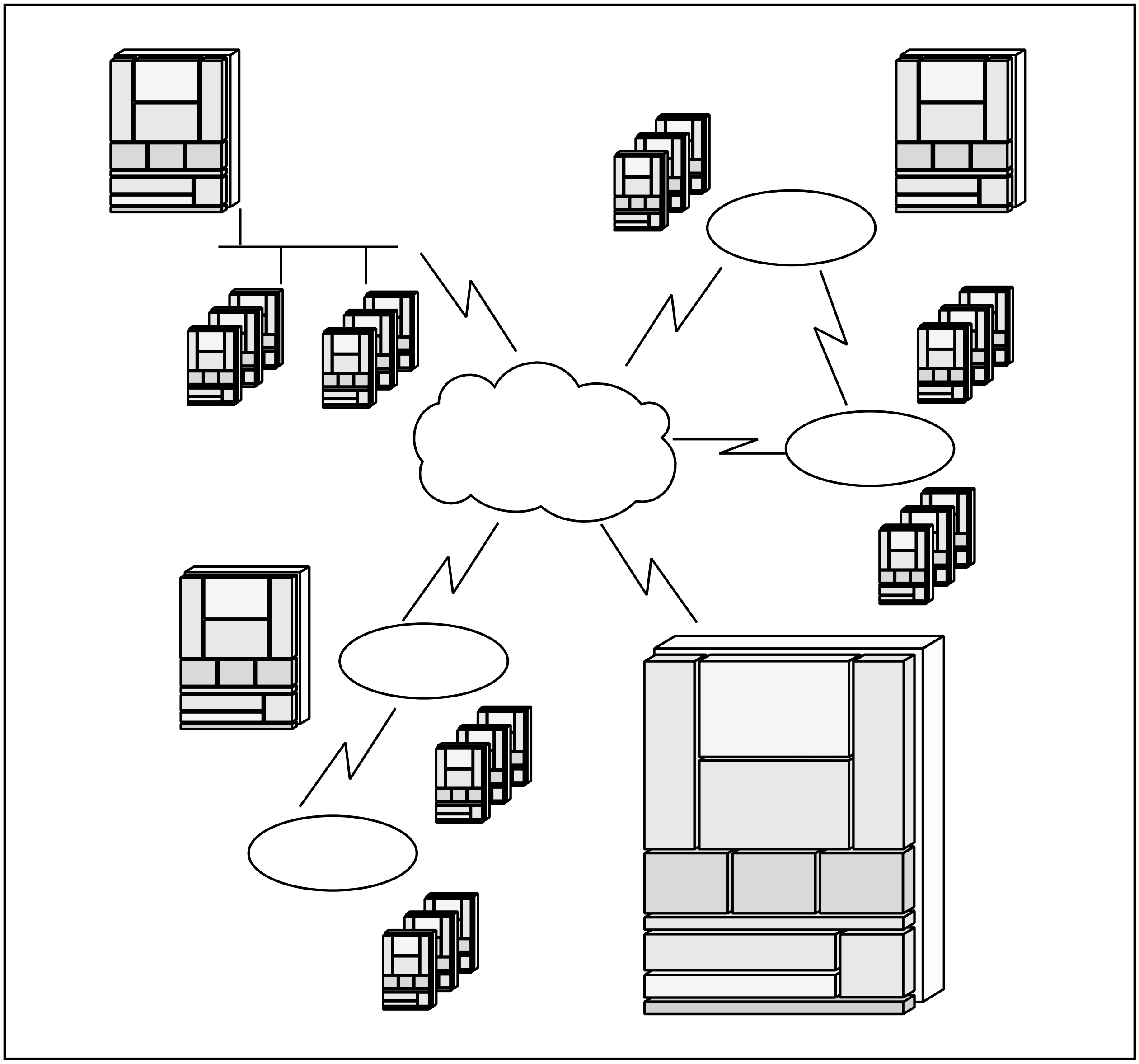

Figure 2. Distributed Operating System

In summary, the Open Blueprint is a structure that will help IBM and others deliver integrated, interoperable products and solutions:

For **end users**, Open Blueprint integration hides the complexities of the network and makes it appear as a single system.

For **application developers**, standard interfaces enable a single system view of the network and allow for the development of interoperable applications that can run on many platforms.

For **systems administrators**, the Open Blueprint defines a consistent way to manage the network to hide the complexity from both application developers and end users.

Chapter 2. Distributed System Characteristics

In recent years, there have been radical technology driven improvements in the price and performance of computer hardware and software. In the past, there were fewer vendors, and users were comfortable dealing with a small number of suppliers. Today, as a result of these improvements, competition has increased dramatically, and there are many information technology suppliers to choose from. The competition, in turn, lowers costs further and spurs additional technical creativity. Users cannot afford to ignore the opportunity to select the best products and services from competing suppliers.

Personal systems, low cost servers, and cost effective, high speed network products are the base of today's computing environment. Personal systems have allowed the user interface to evolve rich forms of interaction. Highly functional, inexpensive, personal productivity applications used on personal systems operate in a world that is often independent of any central computing facility. This new environment has permitted data stored in personal systems to be shared within a work group and centrally managed data to be accessible to individual users. Virtually all major industry applications under development today use personal systems for end user interaction, and exploit distribution mechanisms for sharing data across both small work groups and entire enterprises. Distributed access to resources (such as files and databases) and services (such as printing and electronic mail) are key elements of these configurations.

The distribution of these facilities, however, presents new challenges in designing and using system components and infrastructure. Configuring network products and the requisite software support among the parts of the distributed environment is difficult and often leads to redundancy. For example, one transport protocol may be used to access a service needed by an application and a different protocol may be required to access another capability. Given the number of competing suppliers of systems and applications, and users motivated by function, price, or performance to select from this wide variety of competitive offerings, it is increasingly likely that systems will become more complex as the function is deployed in a distributed environment.

A coherent, systematic approach to system structure and design challenges is required. It should provide the following:

- Diversified but coherent communications support
- Secure access where users and systems are dispersed
- The ability to ignore distribution where it is appropriate to do so
- The capability for users of the systems to manage them

The purpose of this chapter is to describe an **open, heterogeneous**, **distributed system** that supports distributed and client/server applications. Here are some definitions:

Open
: The system defines interfaces that are standard, relatively stable, and publicly described. Users need to be able to choose elements that make up the system from various suppliers based on cost, performance, functions, packaging, terms and conditions, and other factors.

Heterogeneous
: Users expect to compose systems with offerings from many different vendors at the same time and may change the components that make up their system over time. Also, multiple systems may be connected together forming even more diverse configurations.

Distributed System
: System elements are separated from one another and are connected by a communication network.

The interfaces and specifications in the distributed system permit many different suppliers to develop components that work with each other. Many of the specifications chosen are the result of IBM's joint work with other companies and industry consortia. Where appropriate, the facilities of the distributed system support heterogeneity by allowing choices among a variety of mechanisms that accomplish similar functions.

The distributed system accommodates an evolution from existing system structures and product implementations.

Primary Characteristics

Many of the characteristics of an open, heterogeneous, distributed system are essentially the same as those of any computing system. Appropriate performance for applications, high reliability, a flexible configuration, and system integrity are all desirable attributes of any system. But the nature of the open, distributed system puts emphasis on certain characteristics that affect its fundamental design and specific implementation details. The system must be **accessible**, **transparent**, **scalable**, and **manageable**. Here are some definitions:

Accessible Allows access to information and functions from anywhere in the distributed system.

Transparent Masks the complexity of individual functions from their users. Functions support well-defined, simple, functional interfaces. Implementation details are not apparent.

Scalable Accommodates a wide variety of system sizes, from very small work groups to international networks.

Manageable Permits the effective administration and control of all the elements of the distributed system.

Accessibility

The User's View: Users of a distributed system expect to see it as a single, coherent computing facility. They use the computing services through the different hardware platforms and software environments that they have and to which they are accustomed. The users of distributed systems need not concern themselves with where or how services are provided, as the differences in the systems providing the services are masked from the user.

To the users of the distributed system, there is a single point of access and a unified logon. Users are required to identify themselves only once to access any of the services they are authorized to use.

Universal Naming: Within the distributed system the resources are accessible to users and to programs through a consistent and well organized naming scheme. The ability to refer unambiguously to resources in the system through a universal naming scheme dramatically improves the system's simplicity. The universal naming scheme is independent of location and method of access.

Transparency

The User's View: End users and their applications do not see the overall complexity of the distributed system. They view the distributed system as an extension of their individual workstations.

The Programming View: In addition to the end user view, all programs in the system can take advantage of the transparent implementation of functions. By using the defined interfaces and functions of other components in the distributed system, programs can limit how much they have to deal with the complexity of the distributed environment and can concentrate on implementing their primary functions.

There are limits to how much transparency can be achieved. There are important instances where facilities need to appear merely as a “gateway” or a “window into another world.” Those paths may represent an existing application function for which the investment in transparency is inappropriate.

The existence of more than one computer in the system cannot be completely hidden from implementers developing distributed programs. It is desirable that the complexity associated with programming in the distributed system be minimized.

Transport Independence: Many different networking mechanisms are used to support the interconnection of distributed systems. Programs in the distributed system are provided with several levels of communication mechanisms and interfaces that insulate them from the protocols or semantics of the particular network transport being used, and that enable the transparent support of multiple networks.

Scalability

Open Ended Design: The notion of large (possibly global) networks demonstrates that the distributed system must address an essentially unlimited scale. It must also successfully scale down to smaller networks while maintaining appropriate performance and costs. A few workstations and a server system on a LAN can constitute a distributed system.

In large networks, each individual workstation may be relatively small due to overall cost considerations. So, the structure must also permit the subsetting of functions for smaller machines.

This kind of scalability requires that there be no major design discontinuities as network size increases. For example, the requirement to change an implementation algorithm or data format due to network growth would constitute such a discontinuity.

Scalable Performance: In order to ensure that the distributed system can scale up effectively, some key design principles are observed.

Algorithms or mechanisms that require **enumeration** of all or most of the elements in the system are avoided. Enumeration means any activities or data that increase in direct proportion to the number of elements in the system. For example, if a particular program or database in the network were required to “know about” all the nodes in the network, the memory space in that system could become a limiting factor on total network size. Similarly, a protocol that requires an event (such as connection or disconnection) at one point in the network to be “broadcast” to all other nodes (or a large fraction of them) could cause network traffic that itself would limit the scale of the network.

To permit expansion of the distributed system, the mechanisms that do tend to grow in some reasonable proportion to the system size are **partitionable.** As the requirement for a service grows, there are two ways to deal with resource capacity requirements. It may be appropriate to divide that service into multiple instances. Also, the services are individually scalable so that they can take advantage of the most powerful computers and subsystems that can be employed. In both cases, additional computing power, storage space, and connectivity can be used to manage the expanded resource demands. Where appropriate, it is possible to take advantage of minicomputers, mainframes, or parallel processors to provide better performance.

The design of distributed services cannot place an undue burden on the individual systems that make up the network. While each system that participates in providing a distributed service will “see” a cost of belonging to the distributed system, this cost must be small. Activity in one part of the distributed system must not create an unmanageable load on parts that are not participating in or benefiting from the activity. For example, scoping rules and limits could be used to prevent a broad directory attribute search from having an impact on many servers in the network.

In many cases, the performance perceived by an end user for a distributed system service must be comparable to that seen for local operations. An example of this is the use of a shared file server in place of a local hard disk. But, in many other cases, the distributed service provided has no local analog. In these cases, performance appropriate to the scope of the service is expected. For example, delivery of electronic mail over a large wide area or public network is expected to take more time than delivery in a LAN based system.

Configuration Flexibility: The distributed system must accommodate a choice of configurations from relatively low cost and few functions to many functions and high reliability and performance. A small LAN based configuration should be possible using small processors, minimum memory, and limited disk space. The distributed system can also accommodate a variety of hardware configurations of the component systems, such as computers configured as specialized servers.

Manageability

Systems management includes the planning, coordination, operation, and support of heterogeneous networks across a user's enterprise. This includes monitoring system functions, scheduling hardware and software changes, configuring system resources, and tracking system problems and resolutions.

Regardless of how systems management applications monitor and control system resources, the possible system administration actions should be available anywhere in the network.

Integrated Management Applications: Integrating systems management tools in large, complex, distributed systems is a challenging task. The system may have many elements overall as well as many different kinds of resources that present their own unique attributes to be managed. Systems management tools use common techniques to record, administer, and present managed system elements while providing specialized logic to support configuration, monitoring, and control of the unique attributes of particular classes of resources.

Common management and administration techniques must be facilitated through the use of integrated end user interfaces. These interfaces must expose the manageable elements of the system in a consistent and usable fashion. The user interfaces provided for systems management are not different from those used by other applications, and it is possible to use other general purpose and user written applications with systems management software.

The use of other components of the distributed system, such as the directory, database, file system services, and the object request broker, enables the integration of systems management functions with each other and with other applications.

Systems management functions in the distributed system are extensible by the user's organization. Appropriate interfaces are provided that allow systems management functions to be augmented by and integrated with user provided software.

Open Management: The systems management applications and functions of a distributed system define or exploit existing, standard mechanisms to support the systems management process.

It is possible for third parties to add systems management applications to an existing system. It is also possible to add new managed objects, which are supported by existing management applications. New systems management applications are not required for general administration of new managed objects, except for aspects of the new objects that are unique.

Administrative Scope: The distributed system does not impose a specific scope of systems management and administration.

Large distributed systems may not be effectively managed from a single operational or administrative focal point. Alternatively, it may not be possible to practically manage each object individually in these large systems. Local interfaces to manage objects are present, and it is also possible to manage systems resources remotely.

In general, it is also not effective to manage the different classes of resources separately. For example, when failures take place, problems that need to be diagnosed and repaired do not typically confine themselves to an individual class of resources. A network failure may affect a distributed database. A file system problem can cause part of the network to overload. Similarly, the configuration of resources frequently requires a cross domain approach to balance communication bandwidth and file system or database server load.

In configurations that support large and diverse organizations or those that encompass multiple organizations, differing management policies and methods are accommodated. Central support, small to mid-size work group administration, and individual user autonomy are all possible simultaneously.

Management Automation: As the size of a networked system increases, the only acceptable solution to many systems management problems is to have systems that are self managing. Individual system elements are as self regulating as possible and do not require any external configuration or control for most normal operation. The sheer size and complexity of the distributed system may make direct human intervention in the complex and time critical system management activities impractical. Intervention is required only to handle unusual conditions, and the number of such conditions is minimal. Varying levels of automation are applied to make real time adjustments to the system. Automation requirements may also be driven in smaller networks by limited systems administrator skills.

Additional automated logic can recognize and respond to exception conditions, failures, and changes in system load. Systems management functions are accessible to programs through systems management programming interfaces to all significant elements. A programming environment is provided so that management applications can be augmented economically and effectively.

Unattended Operation: In some enterprises, there may be small groups of skilled users that possess the expertise necessary to run their part of the distributed system and are willing to accept that responsibility. In most cases there are also large groups of users that have neither the expertise nor the personnel to manage their own environment. The geographically dispersed nature of large scale distributed systems makes “hands on” management and operation of most resources impractical. In some cases, there is a strong requirement to run a subset of the distributed system without any local computing expertise or direct manipulation. In these cases, the tools for managing and controlling the resources function entirely with remote operators, to the extent that no human intervention is required at all.

The User's Role in Systems Management: The potentially large numbers of users in a distributed system make individual participation in systems management activities both desirable and possibly required. Knowledgeable users may want to be able to manage their own workstation hardware and software configurations. Other activities, such as problem reporting and user enrollment, allow (or even require) end users to participate without specific system knowledge. Finally, some users may be managing resources (such as particular file services, databases, or printers) within a limited scope.

Using Distributed System Technology: Not all systems management problems are best solved by developing new tools to address specific needs. Some systems management problems can be dealt with by employing the distributed systems technology itself to eliminate or reduce the scope of the problem.

For example, the large numbers of individual workstations and personal computers employed in distributed systems have multiplied the problem of managing, delivering, installing, and maintaining required software. In the past, distributing software on removable media (diskettes, tapes, or CD-ROMs) for installation on workstations was practical. But as the volume of installed software packages has grown and the classes of users broaden, distributing software this way is not efficient, not cost effective, and cannot be easily tracked by the enterprise. To solve this problem, the distributed system's own communication infrastructure and applications can be employed. New distributed applications that transmit and install or update software electronically on remote workstations can be written. Alternatively, shareable software libraries that make new or updated software available to work groups can be made available using distributed file servers. Additional license management and asset management applications that regulate or account for the installed software can be used to ensure that an organization complies with the contractual obligations involved in distributing a purchased software product.

Additional Characteristics

Security

Most distributed systems are networks of secure and non-secure elements. Further, a system may be used by parties either who do not trust one another or who can assume at most limited trust in others. Full security facilities are available across the distributed system. Mechanisms exist to permit user authentication, secure communications, information integrity and confidentiality, resource access control, security administration, and auditing of security events. It is possible for a system to participate in the distributed system without exposing that system's security to weakness in the network.

Reliability and Availability

Large distributed systems that serve widespread populations must be as continuously available as is business justified. The distributed system is designed to degrade gracefully during failures. When specific equipment failures limit some paths of access to the system, alternate paths may be available. Outages due to service activities (such as failure diagnosis and repair) or planned maintenance can be accommodated, but their scope and duration is limited to avoid significant disruption. Single points of failure of critical system functions are avoided.

Serviceability

In large distributed systems, the hardware and software resources are specifically designed to be serviceable. Built in mechanisms are incorporated for monitoring, problem diagnosis, and repair. In large complex systems it is difficult or impossible to re-create problem situations. The system cannot be stopped to allow for the re-creation, and it is often difficult to determine what sequence and intersection of events lead to a failure. Therefore, system elements are designed to support the capture of such fault data as traces, logs, and dumps at the first point at which problems are detected. Efficient, automated mechanisms are in place to handle captured problem data and to identify previously known problems.

Accounting

The diverse population served by large scale distributed systems often requires that the cost of delivering services needs to be attributed to distinct consumers of the service. Accounting facilities may be as simple as reporting the utilization of a shared resource (for example, which users are using most of the space on the work group's shared file server) or as complex as keeping journals of resource usage by transaction. Accounting records direct and indirect use of system services. The load on a file system caused by a database service that is supporting an application is an example of indirect utilization. The elements of the distributed system maintain sufficient information to correlate processing to specific user activities no matter where the work is performed in the distributed system. The cost and performance impact of accounting are small relative to the cost of providing a service.

Internationalization

A distributed system is likely to be geographically dispersed across regional and international boundaries. Software may have to satisfy the language, culture, and character encoding needs of a wide variety of users. While meeting this need, the software must fit or conform to the individual needs of each user.

Role of the Open Blueprint

The Open Blueprint is the structure that guides IBM and other vendor developers in building systems that have the characteristics described in this chapter.

Chapter 3. Open Blueprint Concepts

The Open Blueprint structure provides support for the execution, development, and management of distributed applications. This is accomplished through resource managers, protocol layering and gateways, platform and standards support, integration, heterogeneity, and extensibility.

Application Focus

Systems and system software are developed and purchased to run **applications**. The Open Blueprint defines an application as the use of information technology to assist in the execution of a business process. The resource management services in the Open Blueprint can be considered part of the application solutions they support. They exist to be reused by enterprise business processes.

Distributed applications execute in or exploit multiple systems to accomplish their functions. A client/server application is a distributed application that is split into a client portion and one or more server portions, in which the client portion drives the application and typically supports end user interaction. The server portion or portions execute the function requested by the client portion and typically reside on different systems. Server systems support multiple client systems requesting multiple functions.

Applications request multiple system services to accomplish their functions. Applications can take advantage of the Open Blueprint by using its APIs to request services. The APIs simplify application development and maintenance because they mask the complexities of infrastructure "plumbing" as well as the differences between platforms. Applications are protected from changes to the underlying structure and can use technologies as they are supported by the APIs, both procedural and object-oriented. The Open Blueprint includes a Virtual Machine resource manager that provides the industry standard object-oriented Java execution environment. Object-oriented development enables productivity gains in application development resulting from large scale reuse and the ability of objects to directly model the business environment.

Some applications are divided into portions called transactions. A **transaction** may be more complex than is apparent to the end user. A transaction may involve many interactions with a variety of resource managers. In addition, while a transaction for one user is using a distributed resource, such as data, transactions for other users may concurrently use the same distributed resource.

Applications can be designed to support communications, collaboration, and cooperation among groups of users. This may range from situations where information is distributed in an ad hoc fashion to more highly structured processes where the steps are predefined and deterministic. The resultant applications may be passive, in that they leave control in the hands of the user or workgroup, or active in that they play a more proactive or directive role by controlling the flow of group work.

Application requests for resource manager functions do not have to "cascade" through the layers shown in the Open Blueprint diagram (see Figure 1 on page 2). Any resource manager interface can be invoked directly by any application. Using "higher level" interfaces provides freedom from dependence on lower level implementation details.

Applications that use lower level functions should be written such that they support functions provided by the resource managers that they bypass, where those functions apply. For example, communication resource managers pass certain environment state information between the systems in the distributed system. Applications can use the higher level services to avoid this complex functional obligation.

Many industrial strength programs provided by users, vendors, and IBM that are often termed "applications" are, structurally, resource managers. Elements of many resource managers, such as the part that provides the direct interface to humans, are termed "application programs."

Resource Managers

The **resource manager** is the principal structuring element of the Open Blueprint. Resource management is a logical concept. Thus, a specific resource manager should be thought of as a set of programs that maintains the state of a set of resources. Resource managers provide a set of formal interfaces through which operations may be performed on their resources. Resource managers support distribution with separable support for client and server functions. Only resource managers can directly access the resources they control; that is, they **encapsulate** access to their resources. Resource managers request services from other resource managers through their functional interfaces. In order to properly integrate, resource managers use Open Blueprint networking, security and directory, and object management services.

Resources may be distributed and replicated across many systems in the network. A file system, print server, and database manager are examples of typical resource managers.

Figure 3 depicts a schematic representation of a resource manager.

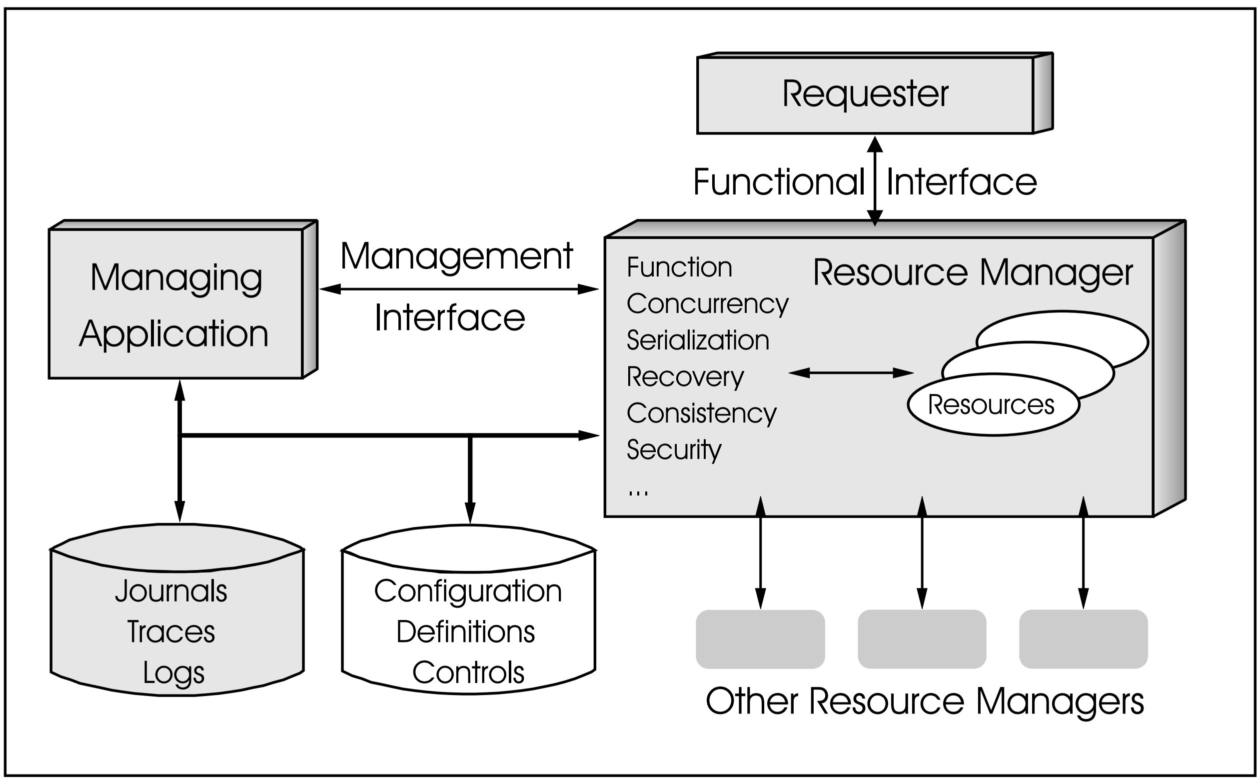

Figure 3. Resource Manager Characteristics

Resource Manager Interfaces

Resource managers provide programming interfaces for the operation, control, and administration of their resources. Programming interfaces support all required resource manager capabilities. There are no operations that require human intervention.

Functional interfaces are either application programming interfaces or protocol boundaries.

- **Application Programming Interfaces (APIs)** are well defined and portable interfaces that are used by user and vendor written application programs. They are also used by other resource managers[2].
- **Protocol boundaries** are well defined interfaces for which only the operations and information passed in the interface are defined, and in which the syntax is implementation dependent. Protocol boundaries are generally supported where performance demands override portability requirements.

In the Open Blueprint, the resource manager interfaces can be structured as a **framework**. Frameworks provide a mapping from the defined functional interface (an API or protocol boundary) to a **Service Provider Interface (SPI)**. The framework permits a particular implementation of a resource manager to be replaced without changes to the programs that use it. Frameworks support heterogeneity by allowing different implementations of resource managers that support the same service provider interface to exist at the same time.

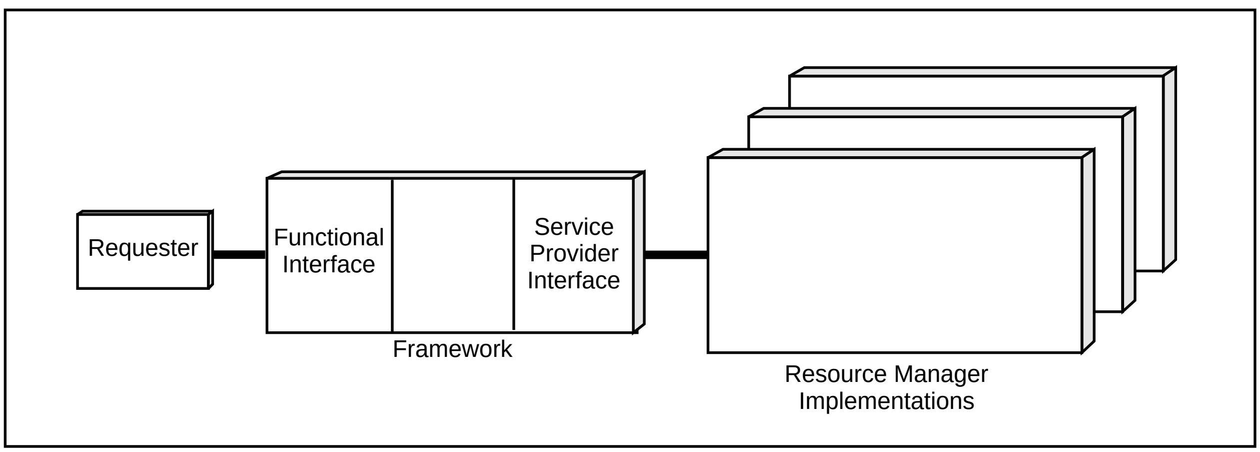

Figure 4. Resource Manager Interface Frameworks

The service provider interface defined by a framework should be a distributed interface so that the service providers can be in different systems than the requesters. This makes it possible to support configurations where a minimum amount of function is required on a small, lightweight hardware platform.

In the case of object-oriented frameworks, a group of object classes provide both API and SPI functionality. The service provider interfaces allow enhancements to (up to complete replacement of) the resource manager functionality offered through the framework. The object-oriented framework may also provide all the resource manager functions. When frameworks are implemented in object-oriented technology, inheritance and polymorphism provide richer methods of customization.

[2] Resource manager functional interfaces are sometimes called **system programming interfaces** when they are designed to be used primarily by other resource managers.

Resource Managers and Systems Management

Resource managers support management functions by:

- Defining their management functions and externalizing those functions through the management interface, so an external entity can monitor and control their function
- Exploiting the common management services of systems management

In the systems management structure, the resources managed by a resource manager are termed managed resources. A resource manager, itself, is a managed resource. Management operations on resource managers include initialization and termination, restart, work prioritization and control, accounting, problem determination, tracing, configuration management, and performance tuning.

Some of these operations require information that does not flow through the management interface, such as journals, logs, trace files, and configuration tables, but may use other Systems Management services. See “Systems Management” on page 70 for more information.

Resource Manager Distribution Support

A resource manager that operates in a single system is a **local resource manager**. A **distributed resource manager** operates across multiple systems. Distributed resource managers include parts that support the interface that requesters use, called the **client parts**, and parts that perform functions on resources, called the **server parts**.

The client program may do some of the processing of the request (for example, validation) and is responsible for determining what instance of the resource manager's server code should process the request. The client program supports multiple protocols as needed to deal with a heterogeneous environment.

The client and server communicate through an agreed protocol, using one of the interprocess communication schemes or the distributed Object Management Services supported by the Open Blueprint.

A server program of a resource manager may process a request entirely by itself, or it may transparently access other instances of its server program through a server-to-server protocol.

Figure 5 illustrates the structure of an Open Blueprint distributed resource manager.

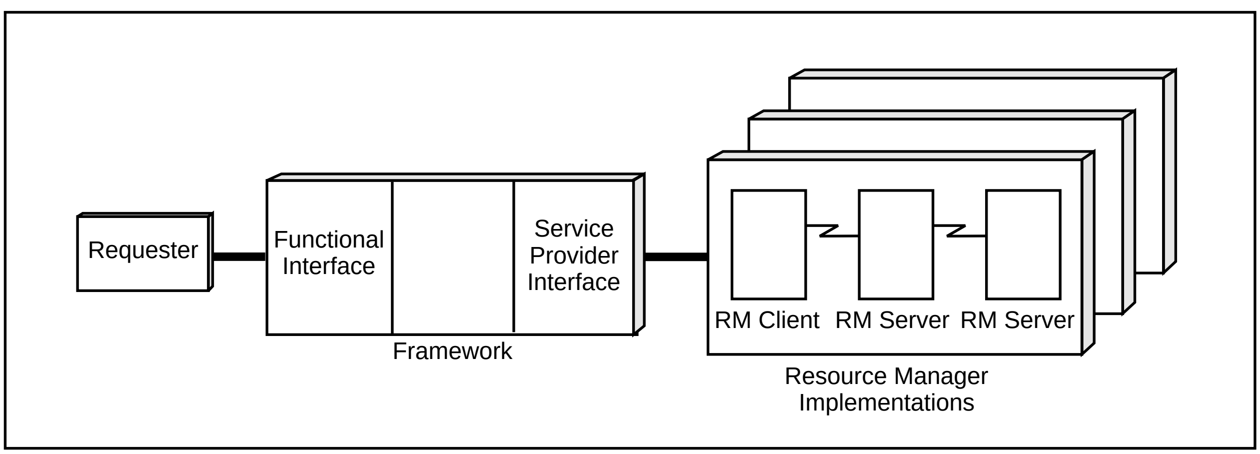

Figure 5. Open Blueprint Distributed Resource Manager Structure

It is expected that different implementers could supply the client and server parts of a resource manager. Thus, functional and systems management protocols are based on standards. Also, each implementation would use the fundamental Open Blueprint facilities that enable integration, interoperability, and a single system image.

Client/Server Terminology

The terms **client** and **server** have both very specific and very general meanings.

In general, the terms client and server describe roles that entities can play when a system performs some work. **Client** refers to the entity on whose behalf the work is done. **Server** refers to the entity that does the work. The terminology is useful when these roles are in different entities, particularly when these entities are physically separated. Often the **server** provides services to multiple **clients**, either simultaneously or serially. This can support shared or multiplexed access by many **clients** to a single resource.

The term **client** not only is used for the “end user” of work but also is applied to any entity that is requesting work. Therefore, a single entity often acts in the **client role** for some work and the **server role** for other work. For example, an application program provides services to a human being (its client) but is itself the client when it requests work of the operating system[3].

In some machines, particularly workstations used directly by people, the client role is dominant. In others, like the machines shared by all the users of a LAN, the server role is dominant. Such machines are often referred to as **client machines** and **server machines**, or **physical clients** and **physical servers**, or, ambiguously, **clients** and **servers**.

Resource Manager Characteristics

Resource managers typically maintain state information for requesting applications and other resource managers across invocations. They must maintain the consistency of the resource state information with underlying system state information.

Resource managers can access environment state information that represents their requesters. They may, if suitably authorized, change the environment state information. Communication resource managers support the distribution of the environment state information by passing it between systems when interprocess communication occurs.

Resource managers support the serialization and concurrency control needed to allow multiple requesters to use their resources. This may require multithreading within the resource manager.

Resource managers manage the integrity, consistency, and reliability of their resources, including recovery from physical and logical damage. Recovery from logical damage involves coordination and synchronization with other resource managers through the use of transaction managers.

---

[3] **X Windows Terminology**

X Windows** is a graphic display system that defines services like drawing lines, characters, and windows on a screen. X Windows has a server that provides these “screen drawing” services, and clients that use these services themselves at the request of an application program.

Just as a file server executes where the shared disk is, the X Windows server executes where the display screen is (typically on the end user's workstation). The X Windows **client** and the requesting application are often on the workstation also, but don't need to be. Thus, what is often thought of as the **client** machine contains an **X server**.

Resource managers employ the necessary security mechanisms to protect resources and information from unauthorized use or unintended disclosure. The Access Control resource manager provides this function based on information provided by the Identification and Authentication resource manager. It is the responsibility of the resource manager that owns the resource to determine when the check is to be performed and the granularity of the resource and operation to which the check applies.

A resource manager's responsibility for problem determination includes detection of failures and capture of relevant failure data when a failure first occurs.

Resource Manager Relationships

Resource managers typically depend on other resource managers for the provision of services. There are two types of relationships between resource managers:

- Those that are dictated by the structure because they have structural significance.

  For example, resource managers are required to depend on the directory to present a single namespace to the user, and on the Transaction Manager to present a single scope for a logical unit of work.

- Those that are an implementation convenience and are of no structural significance.

  For example, a particular product implementation of the directory may choose to store its information using the Relational Database resource manager. This is transparent to the rest of the distributed system.

Object-Oriented Technology

Object-oriented technology is inherent in the Open Blueprint. Resource managers provide object-oriented APIs to ensure requesters can access the functions provided. In some cases, the object-oriented API acts as a “wrapper” for a procedurally implemented resource manager. In other cases, resource managers are implemented completely with object-oriented programming technology, although they allow for access by procedural programs.

A rich set of Object Management Services which support both the System Object Model (SOM) and object-oriented programming languages is included in the Open Blueprint. The most fundamental of these is the Object Request Broker (ORB) which supports basic messaging among the methods of different object instances. This functionality provides local/remote transparent, distributed support based on other Open Blueprint services. The distributed support conforms to the Object Management Group's Common Object Request Broker Architecture (CORBA).

Object management services also include Externalization, Life Cycle and Collections services in support of object-oriented programming.

As objects communicate with each other, certain Open Blueprint services may be executed implicitly as determined by the way the object classes were initially defined. These services are provided by the appropriate resource managers such as Persistence Services, Transaction Manager, and Security.

Protocol Layering and Gateways

Each program in a pair of programs that work together must have some understanding of how the other program operates. That is, they must define the syntax and semantics of the parameters/response passed between them. This definition and its encoding is called the **protocol**.[4]

Protocols are typically nested or contained within each other. For example, the functional protocol used between two arbitrary resource managers is nested inside the communication protocol supported by the Communication Services they have chosen to use. Likewise, the Communication Services protocols are supported on (or within) the transport protocols that are supported by the Network Services resource managers.

Figure 6 illustrates the protocol layering and the need for the protocols to match at each layer.

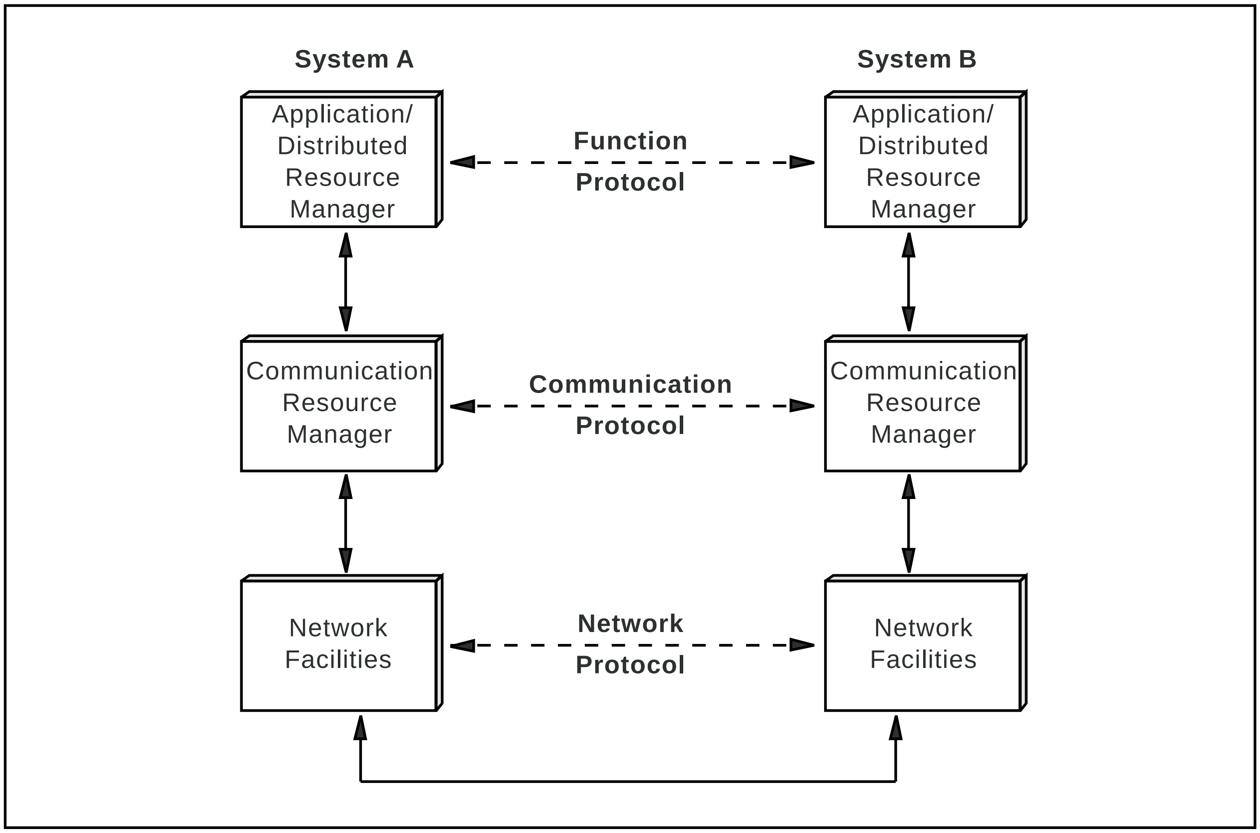

Figure 6. Protocol Layers

Normally, when two programs communicate, they must use common protocols. Sometimes, a gateway is used to support existing programs that provide similar function but do not use a common protocol. Gateways support this interconnection by converting or translating the protocol of each program into the one expected by the other. While gateways can be implemented at any level, a common gateway usage is at the Transport Services level to allow communication across heterogeneous transport stacks (see Figure 7 on page 20). Gateways are a good way to accommodate heterogeneity, but they are not simple and can be expensive. The system that provides the gateway must include both of the transport stacks and the code to do the conversions.

4 The term “protocol” in this book refers to the common architecture usage of “formats and protocols (FAPs)” for the encoding (formats) and the way the encoding is used (protocols).

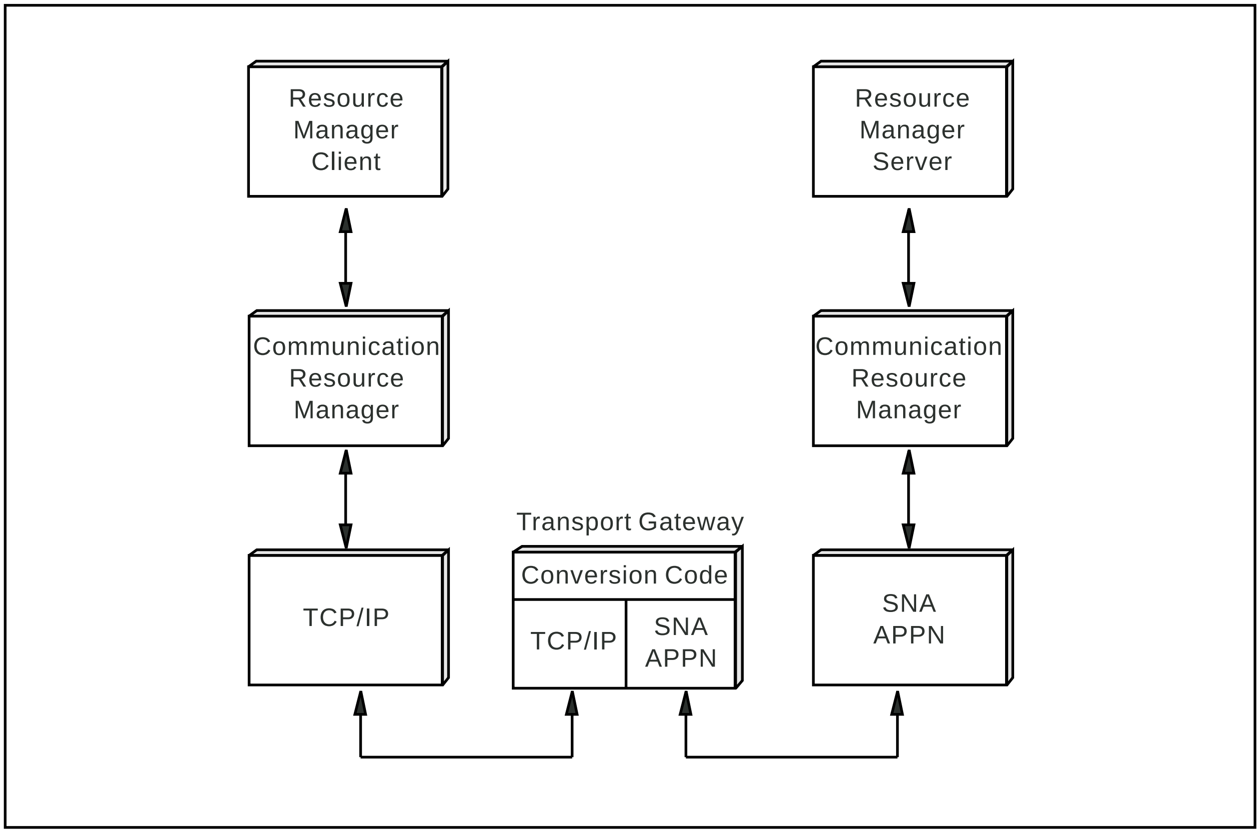

Figure 7. Role of a Transport Gateway

Another common gateway usage is at the Application Enabling Services level, where different relational database functional protocols are converted to support distributed databases. This is discussed further in "Relational Database" on page 62.

Platforms

An operating system and a set of resource managers constitute a platform that is part of the distributed system. Platforms can be configured in many different ways. The Open Blueprint does not demand that any particular set of resource managers be available, beyond that required by technical prerequisites. Packaging and licensing requirements may limit the configuration flexibility, but the following items influence the definition of platforms:

Prerequisite Relationships: Every system in the distributed system must have at least one transport stack, with at least one network driver. For every system it communicates with, it must share a common transport stack or share a common transport stack with another system that contains a transport gateway to the desired end point.

The Communication resource managers have a prerequisite of Transport Services. They use the Directory and Security resource managers. If the Directory and Security resource manager clients used Remote Procedure Call (RPC) to communicate with their respective servers, Directory, Security, and RPC would be a corequisite set.

Most Application/Workgroup Services resource managers have prerequisites of a Communication Services resource manager. Object-oriented resource manager implementations require the Object Request Broker.

Client Platforms: A client system can be fixed or mobile. Depending upon application support requirements, systems can be configured to contain primarily the client parts of distributed resource managers. This would be a typical configuration for an end user workstation. The frameworks for resource managers used by applications must be on the same platform as the applications.

As an alternative, a client system could contain only the Web Browser and Virtual Machine resource managers and their requisite communications support. Even smaller systems, such as personal digital assistants (PDAs), could be configured to contain only a few resource manager frameworks and the communications support.

Server Platforms: Some platforms in the distributed system are likely to be configured as specialized servers. Only the server part of a specific resource manager, such as the File resource manager, together with the client parts of the resource managers it depends upon, would be present. An example of a specialized server is a Web server which is a server containing the server part of the HTTP resource manager. A Web server allows access to applications and other resource managers on that server from requesters on the World Wide Web. For an Open Blueprint distributed system to be functional, the server parts of certain critical resource managers (like Directory and Security) must be accessible somewhere in the network.

Network Computing

Network computing is a form of distributed computing where applications and data reside in an interconnected network. It is the Open Blueprint resource managers distributed throughout the network that actually provide the requested function. From the end users perspective, the notion of specific, discrete servers need not apply.

Network computing can be thought of as an arrangement of function across three tiers. The tiers are logical in that there can be multiple instances of server systems that support each tier, and that the functions of different tiers can be co-resident on the same server system. The tiers are depicted in Figure 8

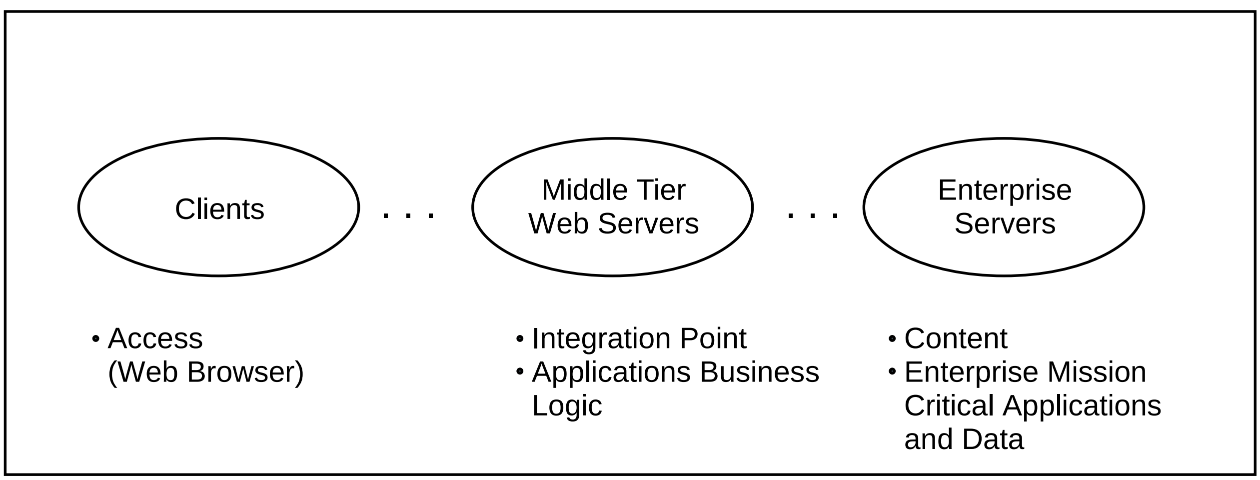

Figure 8. Network Computing Architecture

The tiers are formed by configuring the components of the Open Blueprint appropriately. For example, the Web Browser resource manager provides client access. The HTTP resource manager provides client to middle tier communication and, with its Common Gateway Interface (CGI), access to middle tier function. The Transaction Monitor and Relational Database resource managers provide mission critical applications and data on enterprise servers.

Within the Open Blueprint structure, resource manager servers can be physically distributed across private and public network boundaries. This enables access to services and information from outside the enterprise and permits an enterprise to provide services and information to extra-enterprise requesters.

The most important public network is the Internet. The Web Browser and HTTP resource managers supply specific Internet functionality (or the equivalent intranet function). Additional network computing support includes the Virtual Machine resource manager to support Java, security support for public key technology, LDAP support in the Directory resource manager, and Internet voice support in the Telephony resource manager. The Collaboration resource manager enables many styles of collaborative activities over the network. Many other resource managers support access to and from their resources and usage through the network.

Standards

To meet customer requirements for interoperability, portability, and integration, the components of the Open Blueprint must adhere to standards for:

- APIs that provide a common way to invoke services
- Formats and protocols that allow independently produced software running on the same or different platforms to work together (interoperate)

The interfaces and protocols specified in the Open Blueprint represent IBM's best effort at striking a balance between freedom of choice and consistency. The choice of standards considers the need for customers and vendors to preserve their current software investments, while allowing them to adopt emerging and future technologies.

In the Open Blueprint structure:

- Standards are included from industry consortia (for example, X/Open, Object Management Group (OMG), and Open Software Foundation (OSF)) as well as from formal standards bodies (for example, ANSI, IEEE, and ISO).
- *De facto* and emerging standards are included when there are popular product implementations that provide investment protection and satisfy functional requirements.

IBM will continue to play a leadership role in standards bodies across the world in promoting the adoption of existing standards and the development of new standards where none exist. IBM will also continue to encourage new and innovative ways to accelerate the development of standards.

Standards are included in the Open Blueprint structure because they can provide the function and interoperability demanded by customers. IBM products will provide and exploit standard interfaces and protocols as described in the Open Blueprint structure; IBM products can then be mixed and matched with other vendors' products. In some cases, IBM products will add value by adding function beyond what is addressed by a standard. Further, as new technologies emerge and customers' needs evolve, some standards may be superseded. See Appendix A, “Standards” on page 77 for a list of key documented standards for interfaces and protocols.

Integration

The Open Blueprint structure supports programming development by providing:

- Resource manager definition
- Standards adherence

- Integration objectives and criteria (by using other resource managers and common services)

Many products today implement the standards defined by the Open Blueprint; future products will also implement them. A major goal of the Open Blueprint is **integration** or **seamless interoperability.** Interoperability means that products can work together, usually because they have implemented the same set of standards; seamless interoperability means that the end user does not have to do anything unusual to get the products to work together and has a single system view of the network.

The following scenarios are key focus areas for integration:

Single Signon
Lets the user have a single identification within the network. The "network" could refer to one business or physical network or to multiple businesses and networks. Single signon lets users log on with a single password and have access to all the network facilities for which they are authorized.

Network wide security
Protects network resources and users. It encompasses three basic areas:

- Data encryption to protect data in the network
- Authentication of users and resource managers
- Resource access control to manage what a particular user can do

Network wide directory
Provides information about resources in a network. It eliminates the need for product unique ways to locate resources and shields users from keeping track of where resources are located.

Global Transparent Access
Enables a user to access data or applications in a network or networks without concern for where they reside.

The Open Blueprint structure promotes the integration of a broad range of client/server and network computing products under a single, network wide signon process, a federated directory for locating distributed resources, common services and a graphical user interface for administration. Through the inclusion of a broad set of industry accepted technologies for the underlying directory and security services, the Open Blueprint also promotes interoperability with other vendor provided products.

Structure for Heterogeneity

The Open Blueprint structure supports a heterogeneous environment. It is a structure that divides system functions into distinct components. Each component is required to handle any diversity applicable to its function. Component interfaces are frameworks that are opaque to the variability inside the component. Each resource manager is defined so that it can support code implementing several different protocols that achieve essentially the same function.

The interfaces are also a binding point. A binding mechanism is needed to select the appropriate implementation, depending on the actual circumstance of the interoperation. The directory plays a key role in determining the correct implementations to be bound together. The process for selecting implementations is open, in that the additional implementations to support new protocols are possible by third parties.

These interface frameworks are the key elements of the structure's heterogeneity support. Figure 9 on page 24 emphasizes the support for multiple protocols.

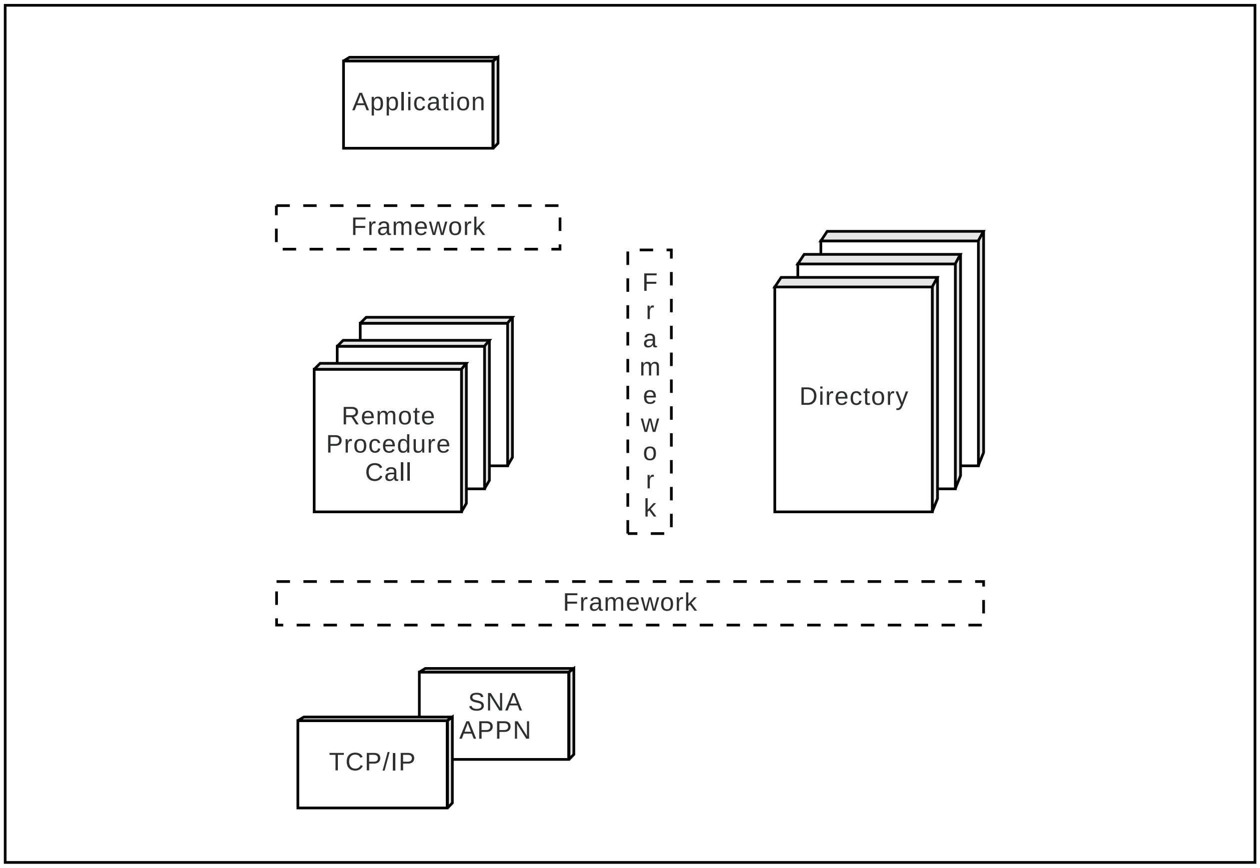

Figure 9. Structure to Support Multiple Protocols

Using such techniques, selecting a single specific protocol for a resource manager becomes only a decision on staging and investment priority. It does not become a commitment to interoperation solely over that single protocol. It does not require universal acceptance of that standard for the system to be successful in the open, heterogeneous, distributed environment.

Extensibility

The Open Blueprint is designed to be extensible along the following dimensions:

- Because resource managers are implemented with opaque interfaces, additional protocols may be supported over time without affecting the invoker of the resource manager.
- New resource managers can be added to the Open Blueprint with minimal effort, exploiting the functions of the existing resource managers. For example, the existing Directory resource manager function is available for use by any new resource manager.
- Because resource managers encapsulate access to their resources, they are replaceable by object-oriented implementations that provide extended function.
- As a result of the Open Blueprint's adherence to standards and well defined modularity, non-IBM resource manager implementations can be incorporated into the Open Blueprint.

Chapter 4. Resource Managers

This chapter describes the Open Blueprint resource managers (as shown in Figure 1 on page 2):

- Network Services
- Distributed Systems Services
- Applications and Application Enabling Services
- Systems Management Services

It also describes local operating system services.

Network Services

Communications and networking are at the heart of the infrastructure for a distributed system. In the more homogeneous world of the 1970s and early 1980s, the communication requirements drove the structure of the application enabling services and subsystems. This typically resulted in tying the enabling services and subsystems to particular communication structures.

In today's world of heterogeneous, distributed computing, the higher level services and resource managers of the distributed system must support multiple operating system platforms and a variety of networking environments. At the same time, the higher level resource managers need program to program communication services that are suitable for their particular distribution model. However, they cannot afford to be tied to specific networking protocols or data link protocols. This led to the structural separation of Communication Services and other transport users from Network Services, as shown in Figure 1 on page 2.

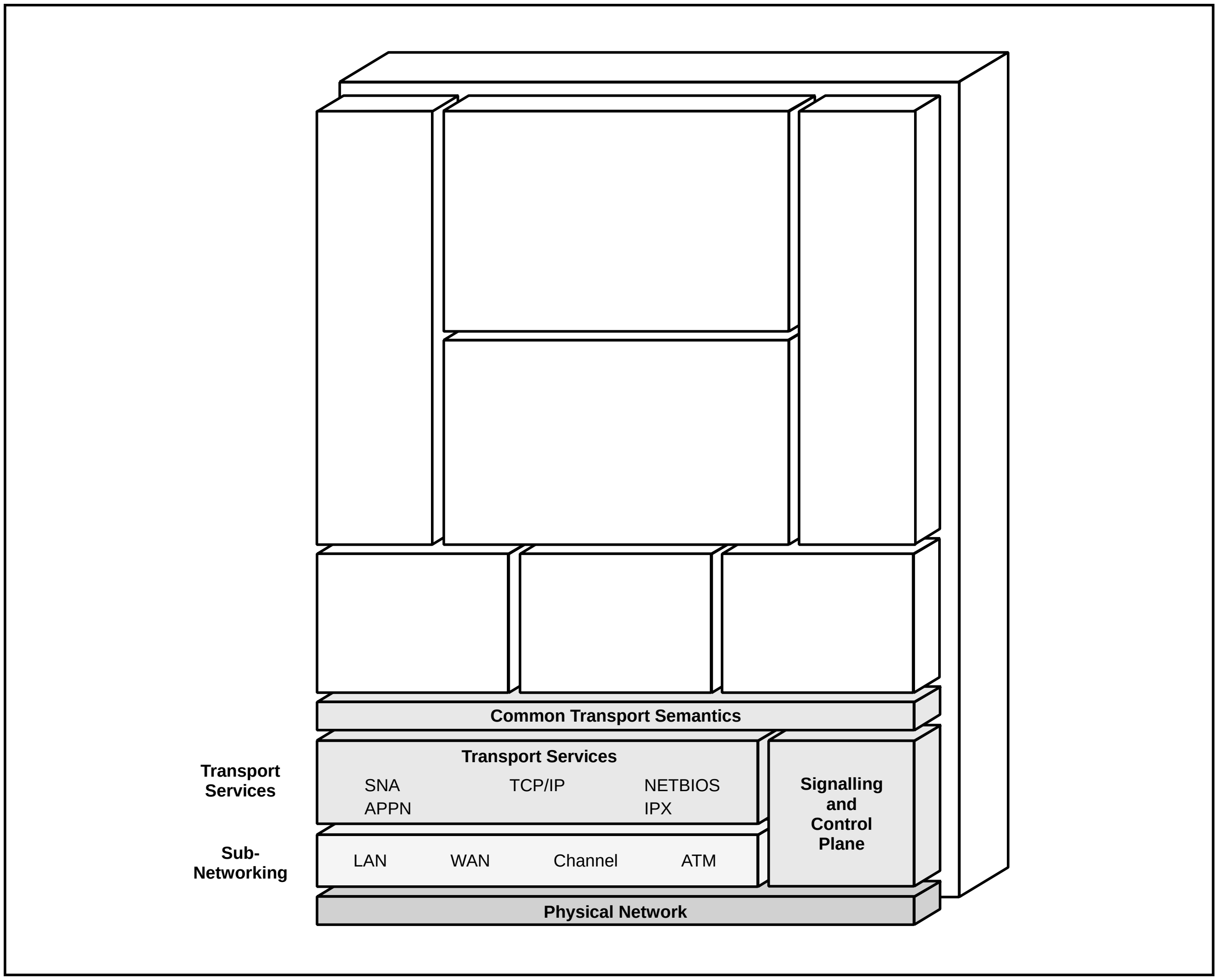

Figure 10. Network Services in the Open Blueprint

The Open Blueprint recognizes the need for structurally unifying Network Services by providing a common view of transport semantics to make higher level distributed systems services and application enabling services independent of the underlying transport network. This leads to the structure shown in Figure 10, in which Network Services consists of Common Transport Semantics, Transport Services, Subnetworking Services and the Signalling and Control Plane.

Common Transport Semantics

Common Transport Semantics (CTS) insulates the higher level services of the Open Blueprint from the underlying transport network (TN) by providing a common view of transport protocols. This common view, coupled with a standard set of compensation mechanisms, enables all higher level services to be transport independent, so that different transport network drivers can be plugged in under a common implementation of those services. New applications and resource managers can be added to any network without adding new equipment or additional communications lines. These applications and resource managers use standard, existing programming interfaces (for example, CPI-C or sockets) without any change.

Using Common Transport Semantics enables the integration of networks with different protocols through transport gateways. As shown in Figure 11 on page 27, gateways provide compensation logic where needed to account for differences in the capabilities of the underlying transport network. For example, this enables the interoperation of client workstations without regard to which LAN media protocol (such as Token Ring or Ethernet) or which LAN transport protocol (such as IPX, NetBIOS, SNA, or TCP/IP) is being used on the particular workstation.

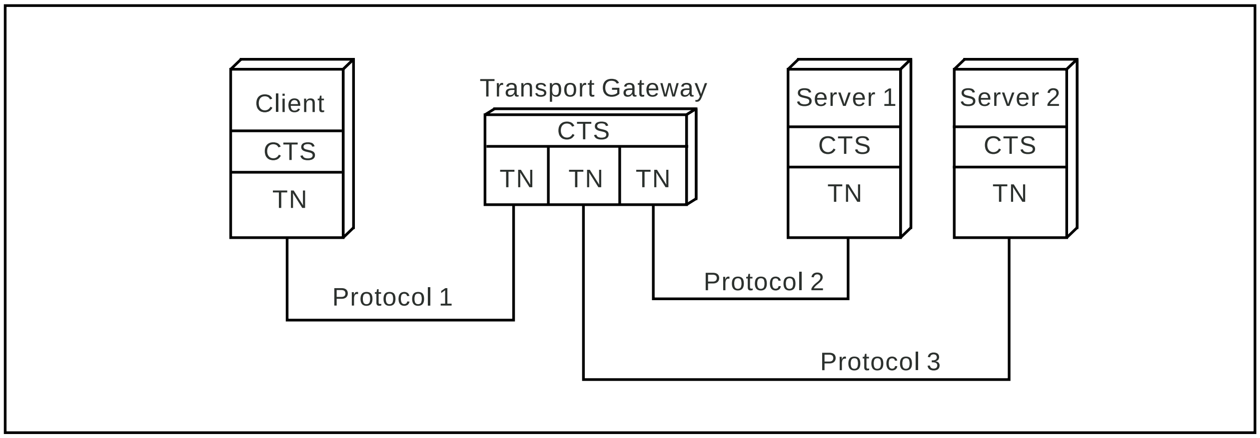

Figure 11. Multiprotocol Transport Gateway

Common Transport Semantics also provides access to the functions of the Signalling and Control Plane.

Common Transport Semantics enables higher level Communication Services (such as RPC or Conversational) to support multiple transport protocols transparently[5]. It also gives applications and higher level resource managers the choice of achieving transport independence by using Common Transport Semantics directly or by using the higher level Communication Services.

The X/Open Transport Interface (XTI) provides access to SNA/APPN, NetBIOS, OSI, and TCP/IP, but it does not shield applications or services from network differences. IBM has developed AnyNet products based on Multiprotocol Transport Network (MPTN) architecture that shield requesters from network differences. X/Open has adopted specifications for MPTN based upon IBM's architecture.

MPTN architecture includes two transport models:

Connection-oriented The transport system is aware of a series of interchanges between the transport end points. It detects and (if possible) corrects lost, out of sequence, or duplicated packets of data.

The connection-oriented model enables the transport system to optimize certain elements and frees the using program from having to deal with certain exceptions (such as detecting out of sequence packets and reordering them).

Connectionless The transport system regards each packet of data as independent, leaving it to the program using the transport system to detect and correct lost, out of sequence, or duplicated packets.

The connectionless[6] model enables "one shot" communication without the overhead of connection setup. It may enable the using program to combine the detection of packet sequence errors with more efficient correction semantics than those which could be applied within the transport system. It also permits other delivery semantics, such as multicast (sending a packet to many receivers).

---

[5] This means that the using system may continue to use its native form of network addressing (such as TCP/IP, SNA or NetBIOS).

[6] Another widely used term is **datagram**. For example, the **Internet datagram** is the basic unit of data transfer under the **Internet Protocol**, and the fundamental Internet service is the **connectionless, packet delivery system**.

The difference between these models is **not** transparent to the program using the interface. The program explicitly uses one model or the other. However, these differences can be masked from applications by their using higher level Application Services or Communication Services.

Transport Services

The Open Blueprint supports a variety of network protocols for transporting information over both wide area and local area networks. These include:

- Transmission Control Protocol/Internet Protocol (TCP/IP)
- Systems Network Architecture/Advanced Peer to Peer Networking (SNA/APPN)
- NetBIOS
- Internet Packet Exchange (IPX)

Each protocol supports interfaces used to access its services. Also included are various end-to-end network monitoring functions that safeguard data integrity and help to avoid congestion. Transport Services plays an important role in supporting end-to-end security in a networking environment. For more information see "Internetworking Considerations" on page 40

Signalling and Control Plane

In a communications network, signalling refers to the collection of procedures used to dynamically establish, maintain, and terminate connections. To do this, information is exchanged among the network users and the switching nodes. For each function performed, the corresponding signalling procedures define the sequence and format of messages exchanged which are specific to the subnetwork (ATM, narrowband ISDN, Public Switched Telephone) across which the exchange takes place.

The Open Blueprint Signalling and Control Plane is based on the International Telecommunication Union-Telecommunication (ITU-T) Integrated Services Digital Network (ISDN) Control plane. But it has been generalized to include the switch connection support for other network types and to include the control for the low level multiplexing of video, audio and data needed by the ITU-T H.Series conferencing standards.

The Signalling and Control Plane is used by the transport services to enable operation over ATM subnetworking and by applications and resource managers requiring the ability to directly establish subnetwork specific connections.

Subnetworking

Subnetworking provides a structure to let networks evolve to accommodate and exploit new high speed, highly reliable transmission technologies without sacrificing business application and network investments.

Subnetworking includes four major types of network connectivity:

- Local Area Network, for example, Ethernet, Token Ring, and FDDI
- Wide Area Network, for example, HDLC, SDLC, X.25, and ISDN
- Channel
- Asynchronous Transfer Mode (ATM)

Each type offers a unique set of configurability, connectivity, and performance options at varying cost levels.

Asynchronous Transfer Mode (ATM) is a subnetworking technology applicable to both local and wide area networks. It is the transfer mode of choice for B-ISDN by the ITU-T, developed jointly with various national standards organizations such as the American National Standards Institute (ANSI) and the European Telecommunications Institute (ETSI).

ATM is a packet oriented switch based and multiplexing technique that uses 53 byte fixed sized cells to transfer information over a B-ISDN network. The combination of small fixed packet size and transmission speed allows support for the integration of voice, data, and video.

ATM can work over a wide range of distances. It is a single common technology for both local area and wide area networks. ATM is scalable in speed from T1 (1.544 Mbps) up to 2488 Mbps. Another aspect of scalability is that ATM does not require the same speed for every connection.

ATM is connection oriented and requires end-to-end connections be established prior to traffic flow. This is done through the use of the Signalling and Control Plane.

Subnetworking is a rapidly changing field, in which major extensions to existing technologies are frequent. In addition, there are a number of rapidly emerging technologies, such as wireless communication facilities (which are key to mobile computing), or cell relay technologies, and very high speed Synchronous Optical Network (SONET) technologies.

The structural separation of subnetworking services and transport services allows customers to separate the choice of transmission technologies from the choice of networking technologies and protocols and to optimize each decision on its own merits.

Distributed Systems Services

Distributed Systems Services in the Open Blueprint provides the Communication Services, Object Management Services, and Distribution Services needed by higher level resource managers to enable the commonly used models of client to server and server to server distribution.

Communication Services

The Communication Services support four common distribution models. Each model describes how distributed parts of applications or resource managers communicate with one another. Each model is supported by a resource manager:

- Conversational
- Remote Procedure Call
- Messaging and Queuing
- HTTP

Conversational Model: In the conversation model, the distributed parts "converse" with one another and are synchronized in a manner similar to the speakers in a telephone conversation. This model is based on the program-to-program communication model defined by CPI-C, where one part of a distributed application or resource manager initiates a conversation with another. The two parts then exchange messages, synchronizing as necessary, until the user's requests are satisfied. Each part of the distributed application or resource manager is responsible for maintaining the state of the conversation and abiding by the rules of the conversation protocol.

The conversational model provides a synchronous service; therefore both parts of the distributed application must be active at the same time. Applications that use the conversational model include distributed transaction processing, distributed relational database access, and bulk data transfer operations involving multiple transmissions.

ISO has chosen the conversational model as the basis for the OSI Transaction Processing protocol specification. X/Open's Common Programming Interface for Communications (CPI-C) provides a common interface for the implementation of the conversational model.

Remote Procedure Call Model: In the RPC model, one part requests a service from the other part and awaits a reply. It provides programmers with a familiar model. Most programming languages support a CALL, and most programmers are familiar with obtaining services by calling routines from a subroutine library. The familiar concept of structuring an application into sets of services and users of the services is extended to a distributed environment.

With RPC, a client program includes a call stub that packages the arguments of the call, sends them to the server program, and waits for a reply. A companion server stub unpacks the arguments, invokes the called procedure, packages the results, and sends a reply back to the client.

RPC has a mechanism for placing (exporting) the definitions of service interfaces into the directory. It includes a mechanism for operation across machines with different architectures, which is supported by the stubs. The stubs themselves are generated by an Interface Definition Language Compiler during the application development process.

The Open Software Foundation (OSF) chose RPC as the fundamental communication model for the Distributed Computing Environment (DCE). The DCE technology for RPC supports connection-oriented and connectionless transports. Because of the richness of the DCE technology, it was selected as the basis for RPC services in the Open Blueprint.

The RPC model is synchronous from the point of view of the calling program because the calling program must wait until the requested procedure finishes its execution and returns the results. Applications include engineering and scientific applications that use RPC to invoke remote, high-performance computing systems, and applications based on the OSF Distributed File System (DFS). Transactional processing applications are also being developed based on transactional extensions to the RPC protocols.

Messaging and Queuing Model: The messaging and queuing model is characterized by distributed applications communicating by exchanging messages. The messages are passed indirectly through message queues which permits independent execution of the partner applications. The communicating applications do not have to be active at the same time. This greatly relieves the application development burden of dealing with network outages.

The sending application uses the Message Queue Interface (MQI) to put a message on a queue. The Messaging and Queuing resource manager takes responsibility for delivering the message to its destination queue, either locally or remotely in the network. The receiving application gets the message from the queue for processing. The applications may choose to use a single queue or separate queues for different message types. Multiple messages can be processed. After processing, the receiving application can generate a reply message to the sending application or forward a message to another application. Distributed applications can be created by arranging the flow of messages between message processing programs. These message processing programs can be reused in different combinations to create new applications.

Messaging and Queuing provides assured, once-only delivery of messages to the receiving system and can optionally start the receiving application. Messaging and Queuing calls can also be used in transactional programs to coordinate message queue updates with updates to other resources such as databases. The assured delivery allows work to be committed by the sender, without waiting for the receiving application to complete, knowing that the message will be delivered despite system or network failures.

Messaging and Queuing is particularly suited to:

- High volume, networked transaction applications
- Intermittently connected networks such as mobile computers
- Heterogeneous networks with multiple protocols or machine architectures
- Networks spanning time zones where network applications will be available at different times
- Access to enterprise applications from the World Wide Web (WWW)

Selecting a Communication Resource Manager: Although it is likely that almost any distributed function **could** be implemented using any of the above three communication models, some applications or resource manager requirements may fit one style better than the others.

Developers of each application or resource manager must choose among the models in the context of their own requirements. Some of the criteria for choosing among the three models are:

- The need for real time synchronization between the partner programs
- The need to communicate between programs that may not be active simultaneously (for example, due to different operating schedules)
- The need to control the flow of communication and resource synchronization
- The need to keep the communication flow and resource synchronization hidden
- The need to allow the calling program to continue execution (not to be blocked) after the communication is initiated

- The need for request/reply based processing, where the calling program is blocked until it receives a response/reply from the partner program

Some resource managers may determine that one model is well suited for certain functions but not others, leading to the use of multiple models.

Hypertext Transfer Protocol (HTTP): HTTP is connection-oriented. A connection is a transport layer virtual circuit established between two programs. In this case, the connection is between the HTTP resource manager's client and server parts. HTTP defines the message as the basic unit of communication. A message consists of a structured sequence of bytes matching the syntax defined in the HTTP specification.

The Open Blueprint HTTP resource manager supports Hypertext Transfer Protocol communication. This enables Open Blueprint client and server systems to participate in and communicate over the World Wide Web (WWW) and over enterprise intranets. HTTP based communication is fundamental to the growth of the Internet and the World Wide Web.

The objective of a communications connection is to enable the HTTP resource manager client, typically invoked by the Web Browser resource manager, to request a resource from the HTTP resource manager server. The resource is a network data object or service that can be identified by a Universal Resource Locator (URL).

Responses to a request typically consist of HTML, including identification of Java applets, as well as Multipurpose Internet Mail Extension (MIME) data types, and increasingly, Virtual Reality Markup Language (VRML).

HTTP communication usually takes place over TCP/IP connections. This does not preclude HTTP from being implemented using any other protocol on the Internet, or on other networks. HTTP only presumes a reliable transport; any protocol that provides such guarantees can be used.

On the server side, the Common Gateway Interface (CGI) allows access to programs named in the HTTP messages. The CGI mechanism provides access to a wide variety of function and content in the distributed system. A server system supporting HTTP is often termed a Web server.

Communication Resource Manager Coexistence: Because it is sometimes necessary for an application or resource manager to use multiple communication resource managers, they must coexist. Examples are:

- A client that issues requests for different services, one accessed through RPC and another through conversations
- A server that supports both clients using RPC and clients using conversations
- A server that is invoked using messaging and queuing, and uses RPC during performance of the service
- A server that is invoked through the HTTP CGI Interface and uses conversations during the performance of it services.

In another form of coexistence, the CGI mechanism can be used to provide Web browser access to messaging and queueing services. This form of coexistence can link the Web browser's user to message processing programs.

Object Management Services

Objects provide a way to create parts of applications by associating data with the programs required to access and maintain the data. These self contained units (objects) afford unique opportunities for development efficiency through improved flexibility and reuse of the implementation. In a distributed context, objects are useful units for portability and transparent distribution.

In object technology, basic mechanisms support the actions of a using program invoking an operation on a target object. These mechanisms support the method call in such a manner that a number of technical characteristics of object technology, such as encapsulation, inheritance, and polymorphism, are supported.

Communications between objects can take place within a process (address space), between two processes in the same system, or between two processes running in separate systems. Objects in different processes use the Object Request Broker (ORB) to communicate. Objects written in different languages that are executing within a process use the IBM System Object Model (SOM).

Object Request Broker (ORB): The Open Blueprint ORB (also known as Distributed SOM) supports inter-process object communication. The ORB enables object distribution by brokering method invocations to remote objects across processes or machine boundaries. The ORB supports a language neutral object model. The ORB enables object method invocations across different language environments. The ORB enables release-to-release binary compatibility so that parts of an application can be updated independently. Using the ORB, objects can be accessed independent of their location, across processes or across distributed systems. The ORB utilizes many services in the Open Blueprint structure including security, directory, and communication services, to accomplish its function. The ORB implements the OMG Internet Inter-ORB protocol (IIOP) as a function protocol to support method invocation across a distributed system.

The ORB resource manager is an implementation of the OMG Common Object Request Broker Architecture (CORBA). CORBA is a standard model defined by the Object Management Group (OMG) and published by X/Open. CORBA defines the ORB, Common Object Services, and the Common Facilities required for supporting objects in a distributed environment.

System Object Model (SOM): When communication between objects developed using different source languages is required to take place within a process, SOM (IBM System Object Model) provides the object request broker function. SOM supports a language neutral object model. It enables in-process object method invocations across different language environments and enables release-to-release binary compatibility within a process. SOM supports C++, Java, and Smalltalk. Other languages are to be added in the future.

A variety of system and language vendors are working with IBM to provide implementations across a range of operating system platforms.

Basic Object Services: A set of basic object services has been defined by OMG. These object services are essential in the SOM environment for application programmers. The basic object services include Life Cycle, Externalization, and Collections services. These terms are defined as follows:

Life Cycle supports object creation, deletion, move and copy. Object creation defines how objects come into being, including the creation of instances and references for objects. Object deletion causes an object to cease to exist. Object copy causes a new instance of the same object to be created, and object move causes the location of an object to be changed.

Externalization provides functions for the transformation of an object into a form suitable for storage or transmission and transformation back into an object. This service is commonly used to provide a means through which objects can be packaged and transported.

Collections provides basic capabilities for building a collection of objects such as sets, queues, and lists.

Distribution Services

Naming:

A major goal of the Open Blueprint is to achieve a seamless, single system image across a heterogeneous collection of systems. To accomplish this, a consistent approach to naming must be established across all resources of the distributed system.

Existing operating systems frequently define one or more namespaces unique to the system. These namespaces can be defined by operating system convention or shaped by specific resource managers. In an open, heterogeneous, distributed environment, the potentially large numbers of resource types and implementations can create a complex array of naming conventions with unique syntax and approaches to context.

Federated Naming: Federated naming is the aggregation of autonomous naming systems that cooperate to support name resolution of composite names through a standard interface. A federated namespace is the logical union of one or more namespaces. To simplify the tasks of using, administering, and writing applications in an open, heterogeneous, distributed environment, the Open Blueprint includes a federated name space based on X/Open Federated Naming concepts and the Internet Engineering Task Force (IETF) Lightweight Directory Access Protocol (LDAP). With federated naming, resources of any class, such as programs, hardware, data, and users, in any location can be referred to by a name that follows a standard set of naming rules.

The federated namespace includes two classes: global and enterprise. Resource manager namespaces may be federated in the enterprise namespace. This allows a resource manager's resources to appear as part of the federated namespace via a federated name.

There are two forms of names:

- Universal Resource Locators (URLs)

  The LDAP URL defines a scheme and uses LDAP distinguished names.

- Federated Names

  Composition of federated names is defined by IETF LDAP distinguished names and X/Open federated names.

Figure 12 on page 35 provides examples of URLs and federated names to show the two different forms of federated names that include equivalent information.

Things to Name	URL	Federated Names
Web page	http://<DNShostname>/<path>	<path>
Distributed file services file	dfs://<cell>/<path>	/.../<cell>/<path>
NFS served file	nfs://<remotehost>/<path>	/<localmount>/<path>
LDAP directory entry	ldap://<hostname>/cn=John Q. Public, ou=Marketing, o=IBM, c=US	c=US, o=IBM, ou=Marketing, cn=John Q. Public
X.500 directory entry	ldap://<hostname>/cn=John Q. Public, ou=Marketing, o=IBM, c=US	/.../c=US/o=IBM/ou=Marketing/ cn=John Q. Public
DCE CDS entry	ldap://<cell>/<path>	/.../<cell>/<path>
DCE Registry entry	ldap://<hostname>/principal=johnq, ou=cellname, o=IBM, c=US	/.../<cell>/sec/principal/johnq

Figure 12. Naming Examples

Over time, new resource managers, APIs, user interfaces, tools, and application programs will support the federated namespace. Existing system and resource manager defined namespaces must, of course, continue to be supported. Resources may be referenced either by their federated name or through a resource manager defined name.

This approach brings the use of existing and new resources which may reside in disparate namespaces into the federated namespace. There is no need for data residing in one namespace to be moved to another namespace for access. Federated naming defines a common global root for all resource identification. Without this concept, specific knowledge of how to find the root of each resource manager namespace is needed to locate all resources. With a common global root, it is possible to build universal resource browsers that can "discover" a resource, even though the browser had no prior knowledge of its type or existence.

Contextual Names: For practical purposes, contextual names are supported. A contextual name is a name that represents some portion of a federated namespace. Contextual names are valid only in a context in which the remainder of the federated name has been provided. How context is established is currently an implementation choice of each resource manager. For existing resource managers, contextual names are often used to map their current programming interfaces to the federated namespace. For example, some portion of a resource manager specific name (such as a device name, disk name, or group name) can be treated as a symbolic reference to an environment variable that contains the remainder of the federated name.

Directory: The Open Blueprint Directory provides a database of information about resources in the distributed system. Because resources in the distributed system follow standard naming conventions, passing these names to the directory services allows resource manager client functions to learn about the location of a resource and all information needed to interact with the responsible resource manager server.

The Open Blueprint Directory reduces the need for "side files" and other statically defined configuration records. The directory enables a resource manager server, at the time a resource is created, to export information about that resource. Client functions can then dynamically learn the location of and how to access a resource, without the resource having been previously defined.

The Open Blueprint Directory is based on X/Open Federated Naming concepts and IETF LDAP. The Open Blueprint Directory resource manager supports a federated namespace. Each namespace can be implemented by a different directory service. A federated namespace allows the data in a particular part of the namespace, backed by its specific directory service implementation (such as that provided by a resource manager) to remain in that place. This allows new directory services to be added and/or existing directory services to be replaced or deleted over time.

The Directory resource manager database (federated namespace) consists of a hierarchical set of names, the namespace, which have associated attributes. Given a name, its associated attributes can be looked up in the directory. Given an attribute or set of attributes, a name or set of names that satisfies those attributes may be returned. For example, given the name of a print server, the Directory resource manager can return the printer's location. Or given a print location, the names of print servers for that location may be returned. The Directory resource manager gives distributed system users a well known, central place to store information, which can then be retrieved from anywhere in the distributed system. The contents of the directory can be spread across many systems. As a result, looking up the information associated with a name can involve interactions with multiple directory servers.

The Open Blueprint provides two classes of general directory services:

- Global Directory Service

  The X.500 and the Internet Domain Name System (DNS) directory services.

- Enterprise Directory Service

  The DCE Cell Directory Service (CDS)/Registry, the Lotus Notes Name and Address Book, and the LDAP directory service.

There is, and will continue to be, a multiplicity of directory services used in the Internet and enterprise. These directory services are federated to provide the Open Blueprint Directory resource manager. This federation has a namespace that the constituent directory services support. The LDAP protocol is used to form this federation. See Figure 13 on page 37 which shows the use of LDAP to accomplish the federation.

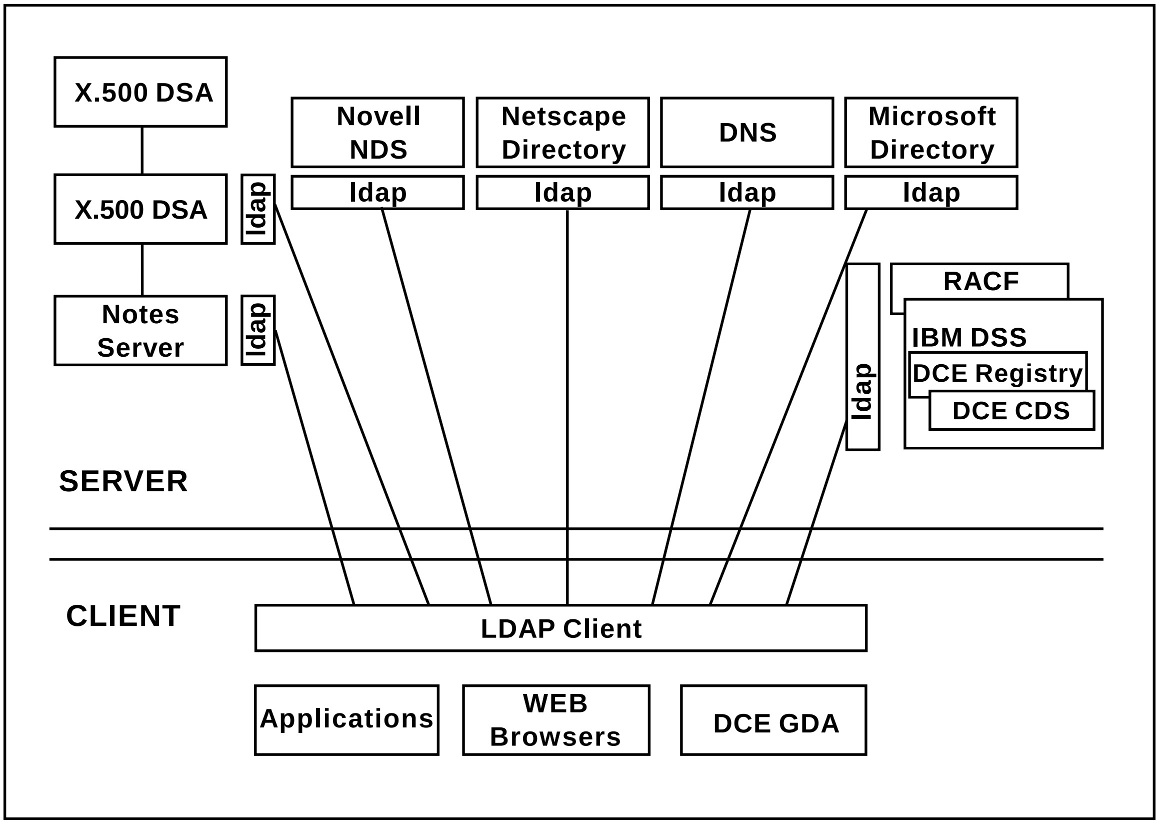

Figure 13. Directory Federation Using LDAP

Because enterprise directory services support the LDAP protocol, and directory services use referral and chaining mechanisms to access other directory services that support the LDAP protocol, a namespace is formed that federates the constituent directory services. The IBM Directory and Security Servers (IBM DSS) are federated in the LDAP namespace. LDAP access to the DCE Registry and CDS allows user and resource information stored in DCE to be brought into the federation. LDAP access to RACF registry information allows RACF stored user and resource information to be brought into the LDAP based directory service federation. Since the DCE Global Directory Agent (GDA) exploits the LDAP protocol for locating other DCE cells, DCE cells are federated into the LDAP namespace. LDAP access to the Lotus Notes Name and Address Book allows information stored in the Name and Address Book to appear as a partition of the directory service federation. Novell's NDS and Microsoft's directory are shown in Figure 13 for completeness based on their respective company press releases.

The API set consists of the LDAP interface (provided though C and Java classes). Each directory service's specific API is still available.

Directory services provide a simple way of naming, finding, accessing, and protecting resources over both space and time. Meta-directory services provide a universal way of naming, finding, accessing, and protecting resources not only over space and time but also across system boundaries as well. A meta-directory service is designed specifically to integrate multiple directories that are connected to it and represents the union of attributes for Meta-directory objects from all directories in an organization. The meta-directory accommodates the selective bi-directional propagation of objects and object attributes to and from any given connected directory. From a management perspective, the meta-directory's primary role is the responsibility for the creation and maintenance of the relationships between objects and attributes across numerous directory namespaces. As a consequence, the meta-directory supports both the meta-directory namespace as well as the namespace associated with each of those connected directories.

Security: Security in the Open Blueprint is responsible for providing the following services to resource managers:

- **Authentication** - Assurance that a user or computer is what it claims to be
- **Integrity** - Assurance that information has not changed in transit from sender to receiver
- **Confidentiality** - Protecting information from unauthorized disclosure or viewing either in storage or during transmission on a network
- **Non-Repudiation** - Assurance that an exchange actually occurred. Both the sender and the receiver agree that the exchange took place
- **Access Control** - Assurance that the user or computer is permitted to do what is being attempted
- **Audit** - Ability to detect attempts (successful and unsuccessful) to compromise security

The above security services are provided by several distributed resource managers:

- The **Identification and Authentication** resource manager provides the authentication service. It also provides the services for obtaining session keys used by the Cryptographic Services resource manager to support integrity, confidentiality, and non-repudiation.
- The **Security Context Management** resource manager maintains and associates transient security state information, obtained by the Identification and Authentication resource manager, with execution contexts (processes and threads).
- The **Identity Mapping and Credential Transformation** resource manager performs a translation between two different security regimes, taking an identity in one regime and returning an equivalent identity in the other. This resource manager provides interoperability between principals that are supported by different security mechanisms.
- The **Certificate Management** resource manager generates, verifies, stores, retrieves, and revokes certificates. A description of a certificate is given below in this section.
- The **Cryptographic Services** resource manager provides all the cryptographic services required to support integrity, confidentiality, and non-repudiation.
- The **Access Control** resource manager provides the access control service.
- The **Audit** resource manager supports the audit service.

The Open Blueprint Identification and Authentication and Cryptographic Services resource managers support both the **secret key** (symmetric key) and **public key** (asymmetric key) security technologies.

As the name suggests, a fundamental characteristic of the secret key technology is a secret, usually a password, that is remembered by a user and stored by a server which resides on a physically secure machine. A user is authenticated by supplying the shared secret, a password, during logon.

The fundamental characteristic of the public key technology is an asymmetrical key pair. The two keys are the *private key*, which is known only by its owner, and the *public key*, which can be universally known. Data encrypted with a private key can only be decrypted with its associated public key. Data encrypted with a public key can only be decrypted with its associated private key.

An electronic document can be encoded with the public key and can only be decoded with the private one. Alternately, the private key can be used to implement digital signatures. In this case, the private key is used to sign, and the public key to verify, the digital signature.

For public keys to be used, an association must be made between the public key and the principal it represents. This association must be trusted. It must not be possible for an attacker to associate a principal with a bogus public key. A trusted association between a principal and its public key is maintained in a **certificate**. The Certificate Management resource manager is responsible for generating, verifying, and revoking certificates.

Authentication: In security terminology, an entity that requests a service is a **principal**. There are various kinds of principals in a system, such as users, servers, and machines. Each principal is assigned an identity that is managed in the distributed system so that it can not be forged and is unique across both space and time. Principals are:

- Initially authenticated as part of the logon process during which their identity can be verified by both a local and network authentication service.
- Authenticated when remote resource managers (for example, a file server) are contacted to perform a service. Typically, authentication takes place during initial session establishment.

When authentication is performed satisfactorily, a principal is assigned **credentials**. Credentials are the set of information in addition to the user's identity that contain:

- Public and private key information
- Secret key information
- Information that can be used by the Access Control resource manager to make access decisions. A user's group membership can be contained in a credential.

When credentials are received from the Identification and Authentication resource manager, they are placed in the local security context of the user. They are available, through the Security Context Management resource manager, to any resource manager client acting on behalf of the user. When this information is transmitted to any resource manager, the resource manager can be assured that it is authentic[7].

Integrity, Confidentiality, and Non-Repudiation: Session keys, obtained during authentication and stored in the local security context of a principal, are used to encrypt and decrypt information that is stored on a machine and that flows between the client and the server. Session keys are used by the Cryptographic Services resource manager to provide information integrity, information confidentiality, and non-repudiation. Information integrity means using matching encrypted algorithmic results to ensure that the information has not been changed. Information confidentiality means the transformation of the information itself through encryption so that only those principals that can provide specific keys can transform the information into a usable form. Non-repudiation allows the receiver of information to verify the identity of the sender of information through the use of an encrypted digital signature. Client code or users are able to specify the options they want to use. These services are provided transparently by Communication resource managers

Access Control: The Access Control resource manager determines if a user is authorized to do the requested function. It compares the authenticated identity and credentials against the access control list that is maintained for the resource. The resource manager through which the access attempt is being made is responsible for initiating an access control check and for faithfully executing the resulting access decision.

Audit: The Audit resource manager provides an audit API that enables the collection of audit records from other resource managers. These other resource managers include audit services that monitor security-relevant system events and principal actions. A set of auditable events is provided from which the administrator can select appropriate activities to be audited. The Audit resource manager supports access

[7] If the system is not content controlled, this information as well as all other information on the system could be compromised. However, the security system has specific safeguards to preclude user passwords or other critical security information from being captured and reused.

to audit records by audit reduction tools that integrate audit records from multiple nodes and resource managers.

Security Administration: Security administration provides centralized administration by which principals can be registered or deregistered simultaneously in a number of domains. A key attribute of security administration is that it can be performed within administrative scopes that are chosen by the organization. The distributed system can contain multiple administrative domains with controlled degrees of trust and delegation among them. Security administration includes functions for authorizing multiple principals to multiple resources following well understood models of centralized administration. In addition, scaling of the registration and authorization functions to a large number of domains as well as resources is accomplished through role based access models. A large number of roles may be created where each role has access to a (possibly) large number of resources which are required to perform the job function associated with the role. Adding new principals is achieved by linking principals to specific predefined roles. Examples of roles are bank tellers, financial credit managers, and loan managers in a banking system. Enterprise roles may be hierarchically related. For example, in a large department store, roles may be sales clerks (by department), department managers, floor managers, store managers, and regional managers.

Protection of the security management information is critical and is achieved by using the identification and authentication, information integrity, information confidentiality and non-repudiation functions.

Internetworking Considerations: Trusted, internal, private networks can be connected to other such networks, sometimes through a secure communications link or intermediate network, but more often through a set of untrusted networks, such as the Internet. Multiple mechanisms exist to provide a secure link through intermediate networks:

- Internet Engineering Task Force (IETF) Internet Protocol Security (IPSec), providing identification and authentication, integrity and privacy at the IP packet level
- Secure Sockets Layer (SSL) Version 3, providing identification and authentication, integrity, and privacy at the sockets interface level

Secure links between trusted networks can be established with hardware, software, or a combination of the two. Most such links depend on cryptographic methods to protect information being exchanged between the trusted networks.

The connection between a trusted network and an untrusted network must also be protected. Firewalls are used as blockades between a trusted, internal, private network (intranet) and another untrusted network or the Internet. The purpose of a firewall is to prevent unwanted or unauthorized communication into or out of the trusted network. Firewall implementations examine network traffic, filtering it and restricting access to system services.

Time: The **Time** resource manager maintains knowledge of the time of day and synchronizes the system clocks in the distributed system to a limited, but known, degree of accuracy. Whenever the accuracy of a local clock's time is beyond acceptable tolerance, time clients solicit the time from several time servers within the network. The local clock is then adjusted based on the intersection of the answers received from the time servers, allowing for processing and network transmission delays. Time servers synchronize with each other so that arbitrarily large networks can be synchronized.

Some time servers, for example, the ES/9000 9037 Sysplex Timer, have access to authoritative **quality time providers** (such as a radio signal), so that even networks that are not interconnected are mutually synchronized[8]. All quality time providers in the world hold consistent time values. Therefore, the Time resource manager would never attempt to adjust the clock of a server connected to a quality time provider.

Adjusting a clock always takes the form of slightly changing its apparent tick rate, so that the clock gradually comes into synchronization, avoiding local requesters observing a discontinuity, and, especially, avoiding the appearance of the clock running backwards.

Time synchronization and adjustment is performed in terms of a time-zone independent time standard, Universal Coordinated Time, (UTC[9]). The Time resource manager also maintains knowledge of human time, which is adjusted for time zones and daylight savings. Human time exhibits discontinuities and runs backwards (at changes between daylight-savings and standard times). UTC has discontinuities only when leap-seconds are adjusted. Time values furnished by humans may be used to determine the offset from UTC to local time but are never sufficiently reliable to determine UTC.

Transaction Manager: The Transaction Manager provides synchronization services so that multiple resource managers can act together to ensure that resources retain their integrity. The resources managed by each of them separately remain consistent according to relationships imposed externally, typically by application programs. The current use of the term **transaction manager** differs from earlier usage. This new terminology has been adopted to accurately reflect technical goals, to accurately reflect the functional parts of the Open Blueprint, and to correlate with standard industry terminology. Major products such as Customer Information Control System (CICS), Encina, and Information Management System (IMS) are combinations of the Transaction Manager, the Transaction Monitor, and other functions.

A distinguishing feature of transaction processing is that all the resource changes associated with a transaction must be committed before the transaction is complete. If there is a failure during execution of the transaction, all of the resource changes must be removed. Resources managed in this manner are called recoverable.

A typical example is a two-part financial application that credits one account and debits another. If a failure occurs after the credit, but before the debit, the application would want to back out the credit. The Transaction Manager interacts with the resource managers involved in the credit and debit, and ensures that either both actions complete or that the accounts remain unaltered. If the two accounts are located in different systems, the Transaction Managers in each system cooperate to eliminate any effects of a failed transaction.

Resource managers handle a sequence of operations against their resources such that the sequence of operations associated with a single transaction either wholly succeed or wholly fail, and, in either case, the state of the resource is well-defined before and after the transaction. It is said that the resource operation sequence is **atomic** and that the **integrity** of the resource is maintained. Thus, a particular resource manager is responsible for the integrity of its resource, which includes recovering from physical or logical damage, backing out incomplete changes, and retrying operations.

Updates to multiple resources need to appear as a single atomic update called a **logical unit of work**. It is this logical unit of work that is the resource managed by the Transaction Manager. The Transaction Manager and resource managers exchange information about a logical unit of work using an identifier called the **logical unit of work identifier**. A logical unit of work is required either to succeed wholly (all updates to all resources were applied successfully) or to fail wholly (none of the updates were applied). Inconsistent states, where some updates have been applied and some not applied, must not persist.

---

8 If they are not synchronized, considerable problems arise if the networks become interconnected.

9 **UTC** is an accepted ISO standard replacing Greenwich Mean Time (**GMT**).

A single logical unit of work consists of operations on resources managed by many resource managers, possibly at several different locations. Several communication models may be used. Services of the Transaction Manager are used to coordinate the atomic completion of the distributed logical unit of work. An instance of the Transaction Manager operates in each system. When activity takes place outside a system, the communication resource manager used is responsible for mediating between the Transaction Managers in each system. The Transaction Manager is not aware of distribution or the type of communication.

Resource managers affected by a logical unit of work register their interest with their local transaction manager and are driven by it through a **two-phase commit protocol**. The two-phase commit protocol is used between a transaction manager and resource managers or other transaction managers to request that the involved resource managers:

1. Prepare to commit the changes (the first phase, which tests each resource manager to make sure that a commit can be performed)
2. Complete the commitment of the changes (the second phase).

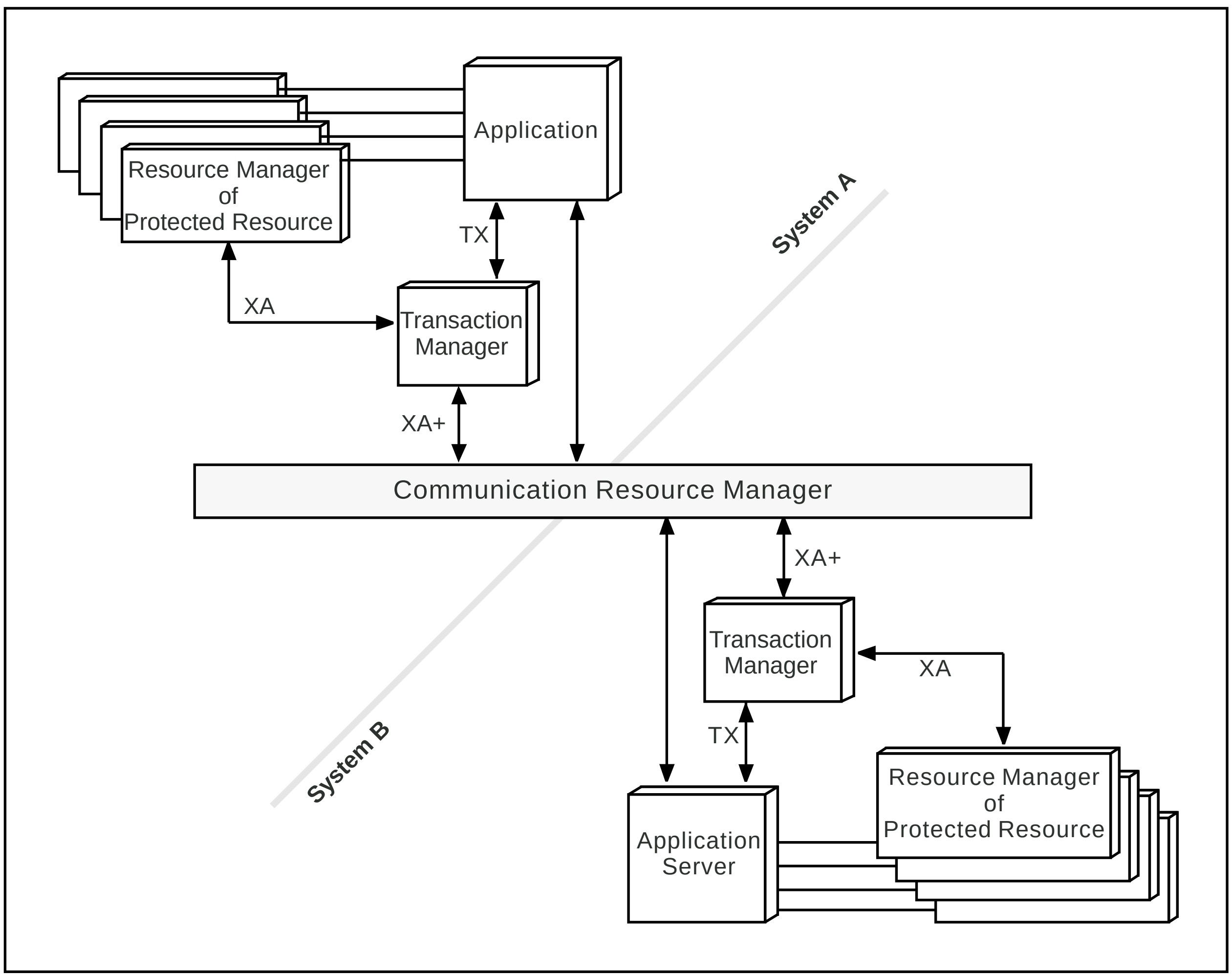

Figure 14. Transaction Manager/Resource Manager Relationships

The Transaction Manager supports both procedural and object-oriented paradigms directly for applications and also as a basis for the Transaction Monitor. As illustrated in Figure 14, there are three procedural interfaces.

TX Interface Used between the application program or its transaction monitor and the transaction manager to indicate when a logical unit of work begins and ends. In addition, if the logical unit of work is ending, the application can indicate whether the completion is correct and the resources can be **committed**, or the completion is in error and the resource changes are to be **backed-out**.

XA Interface Used between the Transaction Manager and the resource managers for the interaction needed to allow the Transaction Manager to synchronize all of the resource changes.

XA+ Interface Used between the Transaction Manager and the Communication resource manager to inform a local transaction manager of the status of a distributed logical unit of work.

In the world of objects, the model (Figure 15) as defined by the OMG, is logically equivalent to the X/Open model with the role of the Transaction Manager defined as the Transaction Service. It might appear to be somewhat simpler because the function of the communication resource manager is handled within the ORB.

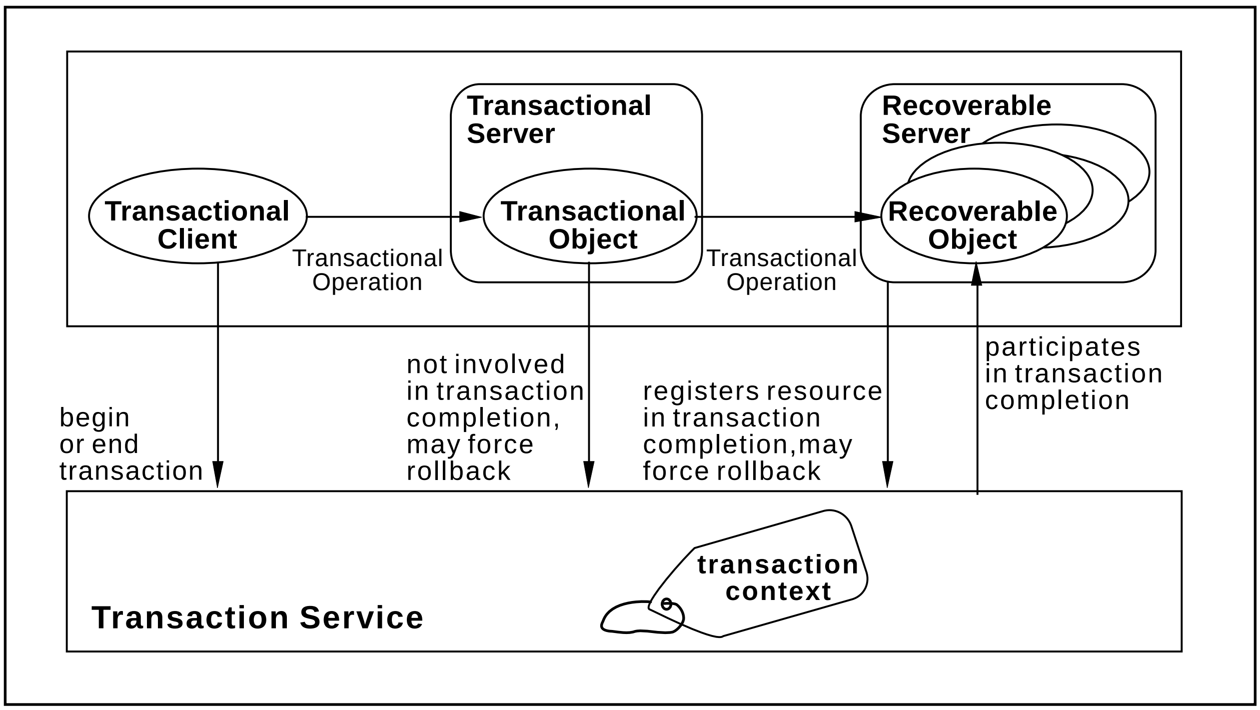

Figure 15. Object Transaction Services Model

A typical application comprises a collection of objects. If an application is transactional, some of its objects will behave transactionally. Such objects belong in one of three categories:

- An object whose behavior can be affected by a transaction but which has no recoverable states or resources associated with it is called a *transactional object*. A collection of one or more transactional objects is called a *transactional server*. Transactional objects and servers do not participate in the completion of a transaction, but they can force the transaction to be rolled-back.

  A transactional server can be likened to the application logic invoked at a remote system using the Communication Manager in the X/Open model.

- A *transactional client* is an object that brackets its method invocations with a series of begin/end requests to the Transaction Service. Within the scope of a transaction, a transactional client can send requests to both transactional and non-transactional objects. Thus, a transactional client is the initiator of a transaction.

- An object that owns and manages recoverable resources or states is called a *recoverable object*. Recoverable objects are transactional objects that have resources to protect and, as such, are affected by the outcome of any transaction in which they participate. A collection of one or more recoverable objects is called a *recoverable server*.

The Transaction Service provides recoverable objects with the transaction coordination services that they need.

Distributed Systems Services Integration

The Distributed Systems Services resource managers work together and with local operating system services to provide an integrated, single system image to using applications and higher level resource managers. Some aspects of achieving this integration are as follows:

Directory At the highest level, information stored in the distributed directory is used by a resource manager to locate the target server (for the requested resource), select the appropriate protocol for communication with the target, and bind the appropriate method or driver code to the logical connection. The process of resolving the federated name of the resource returns the network location (network name) and communication/transport protocol (binding information) necessary for communication with the target server. This information (such as LU name, mode name, and TP name) is then used as input to Network Services to establish the appropriate transport connection and binding and to complete the binding of the required communication method or driver code for the logical connection.

Security Services used to obtain the credentials of the originating user[10] and augment those with credentials for the target server. The full credentials are then passed to the target server with the connection request for the target resource and with subsequent access requests to the target resource, as appropriate, based on the particular Communication resource manager protocol.

Transaction Manager Interfaces with a Communication resource manager whenever communication with another Transaction Manager in a remote system is required to pass synchronization requests for a logical unit of work. The receiving Communication resource manager in turn interfaces with the transaction manager in the remote system to pass the synchronization request.[11]

Local Operating System Services: Resource managers are dependent on a number of local operating system services for support of distribution, including the following:

- Maintaining the identity of the user (referred to as user context or environment state information) on whose behalf the work is being done and providing interfaces for associating the user context with another process or thread on behalf of the resource manager, as required.
- Providing efficient task scheduling facilities for handling inbound communication from many clients and servers in support of resource manager servers of various types.

Applications and resource managers may choose to bypass the Communication Services supplied by the Open Blueprint and implement their own private communication model. If this is done, the application or

[10] This refers to the user on whose behalf the resource is being accessed. Users' credentials are established when they signed on to their systems and the client security services signed on to the network on their behalf.

[11] If the communication resource manager supports a synchronization request protocol, such as the two-phase commit protocol of LU6.2, a mapping between the Transaction Manager protocol and the Communication resource manager protocol is required.

resource manager takes on the responsibilities described previously to provide an integrated, single system image across their using applications and resource managers.

Applications and Application Enabling Services

Presentation Services

Human Computer Interaction: The Human Computer Interaction resource manager, with its associated technologies, supports the presentation of application and operating system information to end users. The overall objective is to do this in an intuitive fashion, allowing the end user to interact with the computer in a very natural, consistent manner. An additional role is to unify and draw from the technology and power of the other resource managers and to abstract away their individual complexities from the user.

The term *Human Computer Interaction (HCI)* describes the front of screen appearance and function of an application or system, along with the mechanisms for the interaction between people and systems. By itself, it does not imply any uniformity between applications. Standards must establish and enforce such conventions.

The term *desktop* conveys a complete metaphor for user interaction. The user's visual space is a desktop. The user's tools and data are things on the desktop or are readily accessible from it. The desktop metaphor has been used in many operating systems throughout the last decade such as the Macintosh's Finder, Windows 95 Desktop, OS/2's Workplace Shell, and AIX's CDE. The state of emerging natural computing technologies (pen, speech, agents, and virtual reality) makes it possible to evolve from the traditional desktop metaphor for user interaction into something more powerful and intuitive. Key to the success of this new generation of the desktop metaphor is that it be a new, compelling metaphor. All components with which a user interacts support customizable, specialized user environments, and the focus is always on end user value.

The Human Computer Interaction resource manager uses the following technologies:

- An abstract component model which allows a component developer to build components which may be targeted to specific component technologies. These include:
  - OpenDoc's[12] compound document technology, including linking and embedding, which enables parts to be interchangeable, shareable, and portable across multiple environments.
  - Object Linking and Embedding (OLE) and Component Object Model (COM) which are of particular relevance to Windows platforms.
- SOM is the underlying, low level object model. It provides the capabilities of binary release-to-release compatibility, multiple language support (in the form of language bindings for APIs), and OMG CORBA compliance.
- Platform specific windowing and graphics systems provide underlying support for the user interface elements.

These technologies, along with technologies such as speech and handwriting recognition, provide content rich, task-oriented environments in which users can be instantly productive.

Specialized desktop environments are based on *places*, which supersede the current, single desktop metaphor. Places are task-oriented. A place contains the resources (including other places) necessary to accomplish a specific task or set of tasks providing a context in which the user works. A place is distinguished by its content, visual appearance, and user interaction paradigms. Existing applications are

[12] OpenDoc is based on specifications of the Component Integration Laboratories (CIL). CIL is a consortium that provides a set of open technologies that support the integration of applications

supported in all places. Places are compound document containers and are able to contain things that are compound document components[13]. Places and the things in them incorporate the latest in natural computing technologies and provide for a rich variety of content.

A component can be shared between different places. Since such places are distinguished by their visual appearance and user interface interactions, it is imperative that the proper separation be made between a component's view and its model (or state). System provided components have such separation.

Places support sharing and collaboration between users. Multiple users can occupy (be "in") a place at the same time. Things in a place can be shared, and users can determine who is in the place and communicate with them.

There is a variety in the manner in which places can represent themselves to a user. This can range from abstract representations similar to today's desktops (using familiar constructs such as windows) to photo realistic places which minimize the use of computer oriented artifacts. Places can be provided for specific activities in the home, such as a library, a place for playing games, and a variety of industry segment specific places. Places can be provided by software vendors, and by users themselves.

Print: The Open Blueprint Print resource manager facilitates print submission, print resource management, and operational tasks in a heterogeneous network environment.

Based on Version 2 of the Palladium print management technology, and developed jointly at MIT with IBM, Digital, and Hewlett-Packard, the Open Blueprint Print resource manager conforms to the International Organization for Standardization (ISO) Document Printing Application (DPA) standard 10175. It tracks the emerging IEEE POSIX 1387.4 standard (formerly the POSIX 1003.7.1 standard), and the X/Open Printing System Interoperability Specification (PSIS) Extensions to ISO DPA 10175 and POSIX 1387.4.

The Print resource manager supports an extensive set of end user functions to submit and control print jobs. Through the client, an end user or application can locate and query printers based on attribute values. The end user can view queues, track the progress of print jobs, and receive notification when jobs have completed or failed. Default job attributes can be specified, supporting the ability to produce consistent printed output, thus eliminating the need to repetitively list certain attributes at job submission.

The Print resource manager provides extensive centralized management and operational control over distributed print resources. Using Print services, an administrative application can manage print system resources like queues, jobs, and printers. It can modify job priorities, monitor the print network, and add, delete or reallocate print resources. Open Blueprint Access Control resource manager services are used to restrict the use of print resources. This enables the implementation of policies governing printer access, printer function, etc.. Open Blueprint Time Services are used to provide clock synchronization, facilitating the use of policies which govern time of day processing (for example, jobs larger than x bytes are restricted to printing during off-shift periods).

The Print resource manager is structured as a client and two forms of server: a **Print Device Supervisor** and a **Print Spooler**.

The client provides an interface to interrogate, manipulate, and modify the printer supervisor and spooler, as well as print jobs. Normally clients work with a Print Server, synchronously delivering the document[14] and job information (for example, scheduling and media specifications) to be printed.

---

[13] This use of the term component relates to objects that the user recognizes and manipulates, e.g. desktop items such as a printer, folder, or mailbox and parts found in a compound document such as a graph, photograph, equation or text.

[14] Or the name of the document. Various options are supported to control whether the requester, the client, the print spooler, or the print supervisor reads the file.

A Print Spooler manages a queue of print jobs, scheduling them to one or more Print Device Supervisors. Within the spooler, a physical printer is represented to an application as one or more logical printers (a collection of printer attributes). This provides the separation between the application's view of a printer and the actual physical printer.

Each instance of the Print Device Supervisor manages one or more physical printers and deals with printing one or more documents at a time. A device support framework provides a consistent set of interfaces for device drivers. This eliminates much of the need to rewrite hardware device drivers for each unique operating system platform. The Print resource manager supports industry standard print data streams, such as PostScript, HPPCL, and AFP.

The client/spooler/supervisor protocol is based on ISO DPA and is implemented using RPC[15]. Line printer controller and spooler/line printer domain (lpr/lpd) input is supported by a function protocol gateway for inbound print requests. Interoperability in a heterogeneous environment is based on an X/Open PSIS implementation.

View: The Open Blueprint View resource manager provides the ability to view, on the workstation, printable documents based on industry standard formats. This permits a document to be viewed before, in lieu of, or after printing on paper, microfiche, or other media.

Documents are viewed in a WYSIWYG (What You See Is What You Get) manner. The document views may be manipulated by changing the orientation and size (scaling) of a page. Text annotation, without changing the original document, is supported using standard desktop editors. Document navigation (moving from point-to-point in a document) is supported using search criteria based on text strings, pages and indexed elements. The index search is limited to documents which support the MO:DCA-P indexing architecture.

The View resource manager supports six separate classes of print data streams. These include PostScript, which has been converted to the Adobe Portable Document Format (PDF); ASCII, while ignoring graphics controls; common fax and scan file formats including bitmap, Tag Image File Format (TIFF), Picture Exchange Format (PCX) and Distributed Control Extension Format (DCX); a common Internet graphics format called Graphics Interchange Format (GIF); the most common AS/400 spool file format, 3812/3816 SNA Character String (SCS); and the de facto industry standard host production printing data stream called Advanced Function Presentation Architecture (AFP), Mixed Object: Document Content Architecture—Presentation (MO:DCA-P).

Multimedia: The term multimedia means two or more media, at least one of which is a time based digital medium: audio, animation, or video. Time based (isochronous) media must meet stringent timing and synchronization constraints when played or recorded; for example, 24 frames per second for movies, 8,000 samples per second for voice. Multimedia exists in one of two states. Either it is already stored on a storage medium or it is being captured "live" in real time. Stored multimedia data may occupy considerable storage space. Ten seconds of compressed, VCR quality digital video requires approximately 1.5 megabytes of storage. Multimedia when played or recorded appears as a continuous stream of data packets, a **data stream**, which must be delivered from a source device to a destination device satisfying stringent end-to-end performance constraints. For example, in a multimedia file server application a performance constraint would be set on the end-to-end transit time of a data-stream packet which would be the sum of the access time of the disk array attached to the file server and the transit times through file server, network, client, and its multimedia adapter at the user interface. In a distributed system, all resource managers involved in providing a multimedia service have to agree to meet specific performance constraints. This agreement is referred to as **guaranteed quality of service (QoS)**.

[15] Print data is transferred using RPC or through an external reference.

Multimedia support allows computer systems to retrieve and present or capture and store distributed graphics, audio, video, and animation data. Because multimedia is digitized, it can also be indexed, searched, and manipulated. In addition, a personal computer that can capture, present, and transmit over a network voice, video and image has the potential of being used in a wide range of communications applications, including telephony, facsimile, multimedia mail, electronic documentation distribution, and audio/video conferencing. Multimedia applications capitalize on the interactivity of a computer and the ability to emulate human modes of communications that makes the computer much easier to use.

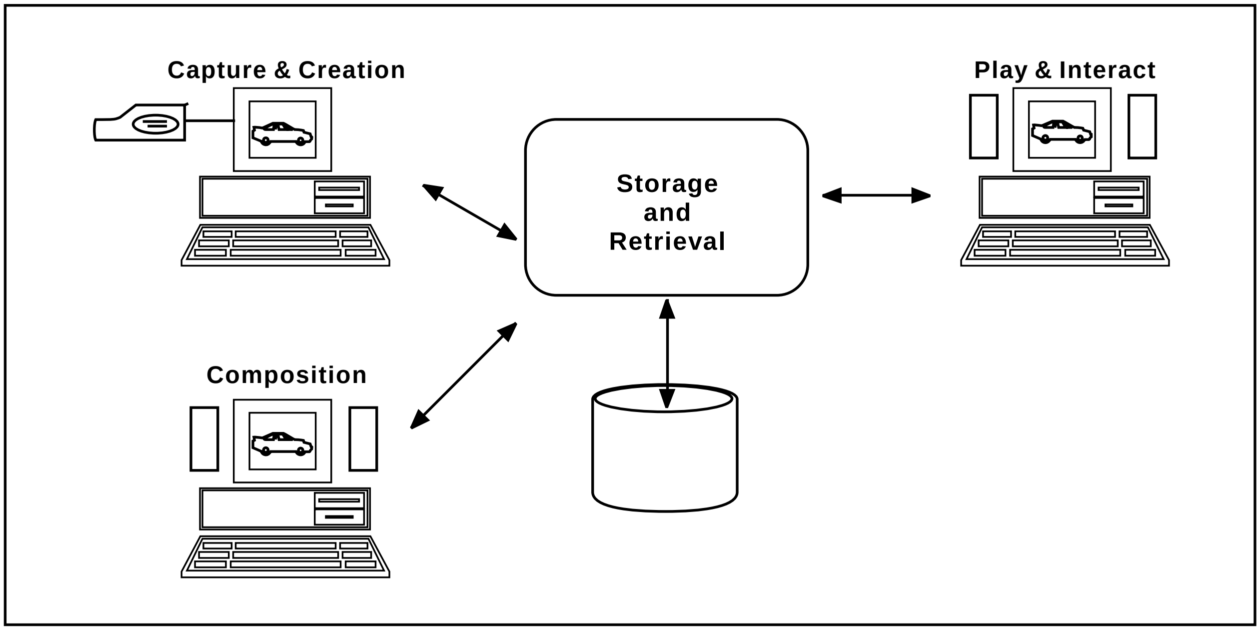

Figure 16. Generic Multimedia Application Capabilities

Figure 16 on this page illustrates generic multimedia application capabilities. Multimedia tools and applications can include capture and creation, composition and editing, and the interactive presentation of multimedia. Multimedia content may be stored in a digital library to facilitate sharing and reuse. Multimedia services can be used in extensions to line of business applications or specialized new applications such as distance learning.

The Multimedia resource manager is composed of core and enhanced services. The core services of the Multimedia resource manager provide the foundation for delivering multimedia data streams to a wide range of applications. Enhanced multimedia services, built on top of the core services, provide functions suited to application domains and network infrastructures.

The core services of the Multimedia resource manager are used to control:

- Streaming functions, which support retrieval and playing of data streams
- Asset management functions, which support content management applications
- System management functions, which are used for administering and monitoring the Multimedia resource manager

The enhanced multimedia services are Internet video services and media streaming services.

Internet video services extend the Multimedia resource manager to provide streaming video and audio to workstations connected to the Internet or to an intranet. Internet video services are integrated into a World Wide Web infrastructure, providing the function in a well defined client/server configuration, and using standard internet protocols. Users can use Internet video services to augment existing Web applications or to build new applications.

Media streaming services also extend the core services of the Multimedia resource manager to support a wide range of applications. Generally, these applications use television monitors to present the data streams rather than personal computers. Media streaming services can deliver digital video streams in analog forms over cable networks.

The decoding and rendering of compressed video and audio streams is done by local system multimedia system services.

Web Browser: Web browsers have become the client interface of choice to Internet computing. As the user's gateway into the World Wide Web (WWW), Web browsers provide a client based end user environment for information delivery and interaction with web servers. The Web Browser resource manager provides the user with a point and click means to browse and navigate Web content without overt knowledge of the underlying servers accessed. The Web Browser resource manager includes capabilities to provide an attractive electronic publishing medium and client interaction based on plug-in support and Java enabled capabilities. Because the Web Browser resource manager can provide distributed presentation support for a wide variety of server based applications, it provides the capabilities of a “thin client”.

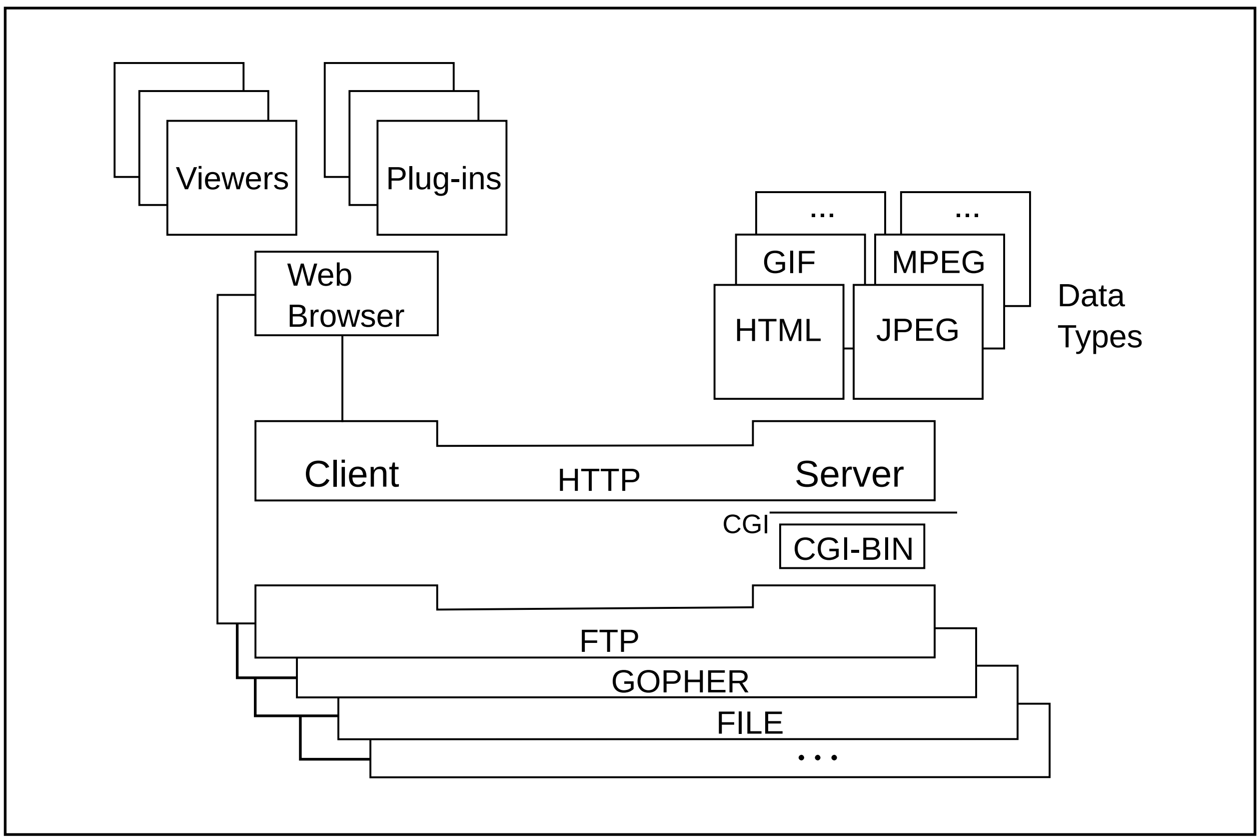

Figure 17. The Web Browser Environment

As shown in Figure 17 the Web Browser resource manager operates primarily through the HTTP resource manager to handle HTML and other data types supported by the HTTP protocol. HTTP (designated by http:// in a URL) is the most common protocol found on the Web. This protocol defines how content such as hypermedia files move from Web server systems to the browser on a client system. However, the browser can support other protocols, such as FTP, Gopher, File, and Telnet.

Functions supported include WWW navigation, forms processing, virtual machine (Java) dynamic content, and support for multiple data types. Browser data types are defined as MIME types. The browser handles some data types itself, for example the MIME types of “text/html” and “image/gif”. To handle other data types, the browser invokes the appropriate helper or “plug-in” application, for example the View resource manager, the Acrobat reader, a Postscript viewer, or the Multimedia resource manager.

Application/Workgroup Services

Transaction Monitor: *Transaction monitor* is an industry term for functions that traditionally were included in IBM's “transaction processing systems,” along with transaction management functions.

The Transaction Monitor provides applications with access to a variety of application and system services (*application servers*). A transaction monitor environment provides a broad set of the necessary elements to manage and run user programs. The Transaction Monitor addresses aspects of program execution, security, system management, and transactional and service integrity.

The major attributes of the Transaction Monitor are as follows:

- **Integrity**

  The Transaction Monitor utilizes the functions of the Transaction Manager (described on page 41) to ensure the integrity of distributed resources by providing a robust, general purpose environment which enforces the Atomicity Consistency Integrity Durability (ACID) principles without requiring complex application code to be written. These ACID principles provide appropriate levels of protection between applications and resource managers and between the applications themselves. This may be accomplished in a tightly-coupled way using a two-phase commit protocol or in a loosely coupled way through the use of transactional queues.

- **Robustness**

  The Transaction Monitor provides an environment in which concurrent applications (or instances of an application) are isolated and protected from one another and in which the business logic is usually separated from both client processing (presentation services) and resource processing (data access).

- **High performance**

  The Transaction Monitor provides high performance by having resources pre-allocated, exploiting early binding to resource managers and other optimizations. A pool of pre-loaded application server processes reduces overall system resource requirements and avoids the overheads associated with starting up a new process for each client request.

  The Transaction Monitor provides efficient use of resources through process management and support for both static and dynamic load balancing. Requests may be prioritized and server processes may be replicated as required, either on the same server node or on different nodes. These facilities are especially important for systems which run on Symmetric Multi-Processing (SMP) hardware.

  Facilities are also provided to allow priority scheduling for differing classes of work.

- **High and continuous availability**

  The Open Blueprint Transaction Monitor handles many different failure scenarios. Because the monitor is at all times aware of the current state of all client/server resources which are under its control, the point of failure can always be detected and failed processes restarted as required.

- **Security**

  The Transaction Monitor extends the functions of Open Blueprint Security by providing additional authorization models. Access to resources can be controlled by any combination of the identity of the user requesting the access, the type of action being requested (transaction-based security), the system or terminal from which the request has been initiated, or the time of day.

- **Scalability**

  The discipline of developing applications for the Transaction Monitor environment is one which leads to the development of modular procedures which separate the function required from the data to be processed. As more functions are added the Transaction Monitor is able to distribute that function over multiple servers. Enforcement of the ACID principles ensures that disparate functions work together in a consistent way.

  In addition, the Transaction Monitor provides an easy way to mix differing resource managers in a heterogeneous environment. Coordination across the resource managers is managed by the Transaction Monitor itself. This allows the environment to grow without requiring any alterations to the existing applications or application architecture.

Thus, the Transaction Monitor provides a ready built framework for running and administering a distributed application.

Currently, there are no formal standards for the Transaction Monitor application programming interfaces. The CICS transaction monitor API has been implemented on all major IBM platforms as well as on several non-IBM platforms such as HP-UX, Sun Solaris, and Digital Unix. The IMS transaction monitor API has been implemented on a variety of platforms supporting applications associated with mainframe systems. The Encina transaction monitor API has been implemented across a range of UNIX platforms and Windows NT.

Event Services: The Event Services resource manager allows a program to send information (representing the occurrence of an event) to another program. The event service decouples the communication between the program generating the information and the program which deals with the event.

Event Services is an implementation of the Object Management Group (OMG) Event Services Specification and provides a simplified interface to the Messaging and Queuing resource manager giving a subset of the Messaging and Queuing function. Events are a one-way communication mechanism with event data generated by an event supplier and passed to event consumers. An event channel, built using a message queue, can be used between the supplier and consumers to allow asynchronous communication. Event suppliers and consumers can be on the same node or distributed in a network.

The passing of the event data may be initiated by either the consumer or supplier. A consumer may pull data from a supplier or a supplier may push data to consumers. A particular supplier application will, most likely, be designed to support either pushing or pulling but could be designed to support both.

Compound Document: The Compound Document resource manager supports a component based programming model in which small task specific software components (or parts) from multiple vendors can be assembled rapidly by programmers or end users to create customized software solutions for a wide variety of business or personal tasks.

The use of the term "compound document" is common in the industry because this kind of component technology was first applied to mixing data types and their editors within a single word processing application. However, the technology is general in its ability to allow components to be integrated for data sharing, data interchange, presentation, and user interaction. Component function is not limited to data editing or entry. Components can update files, monitor physical equipment, perform computations, etc. With the generalization of capabilities, what were once data types have evolved to be more generalized parts, and what were editors have evolved to be described as part handlers.

The major elements of the Compound Document resource manager are:

- Presentation management protocols

- Data interchange format and protocols
- Scripting enablement
- Modeling framework
- Presentation framework

Compound document protocols should be understood to be made up of a set of object interfaces and the semantics of their proper use and interaction. All protocols conform to the CIL OpenDoc specification.

Compound document presentation management protocols ensure that components can be used together in a common presentation to the user. These protocols cover geometry management, human interface event distribution, shared user interface controls, and rendering management.

Compound document data interchange includes a canonical storage format as well as protocols for storage management and data links between parts.

The compound document scripting support is described by the CIL OpenDoc Open Scripting Architecture. Scripting of compound documents is the notion of invoking a part handler's behavior through any OSA conforming scripting language. To improve the ability for parts from multiple vendors to be combined, the OpenDoc specification defines part types which support specified Event Suites of standardized operations.

A compound document modeling framework, which conforms to the Taligent CommonPoint Compound Document Framework specification, provides an extensible implementation of part handlers. This makes it much easier to create parts, and ensures that parts will be well behaved and have an extensible and flexible structure.

The compound document presentation framework conforms to the Taligent CommonPoint Presentation Framework Specification and provides for one or more user interface views to be associated with a part. The Presentation Framework ties the modeling framework to a user interface view. These views are constructed using the Human Computer Interaction resource manager.

Intelligent Agent: Intelligent agents are software entities that carry out some set of operations on behalf of a user (or another program) with some degree of autonomy and, in so doing, employ some knowledge or representation of the user's goals and desires. Intelligent agents can help cope with the flood of information available in the network computing environment and can help simplify user interfaces. Intelligent agents exhibit a degree of reasoning and learned behavior.

The Intelligent Agent resource manager provides functions and facilities that an intelligent agent uses to do its job. Object-oriented technology is the base for the sub-components of the resource manager. Specific intelligent agents are composed of instances of the resource manager sub-components.

The sub components are:

- **Adapters**, the “eyes, ears, and hands” of the intelligent agent. They are the links to the outside world. Adapters are how an intelligent agent interfaces to applications or to other resource managers in the system to sense and effect its environment.
- **Engines**, the “brains” of an intelligent agent. Engines give the agent its initiative and its autonomy. Engines operate using different forms of knowledge.
- **Knowledge**, used by the engines when they operate. Examples are programs and scripts that define agent behavior and sets of rules that encode the preference and intent of users that the agent represents.
- **Libraries**, various forms of knowledge persistent across agent activations. To promote the sharing of rule sets among different inferencing engines, rule sets are stored in the library in Knowledge Interchange Format (KIF) format. KIF is a proposed ANSI X3T2 standard.

- **Views**, tools for the user to interact with the intelligent agent. For example, rules governing the agent's behavior can be examined and modified using a rule editor that is part of the view subcomponent.

The Intelligent Agent resource manager binds the adapters, engine(s), and library that comprise an intelligent agent when it is started. The Intelligent Agent resource manager provides services to the agent while it is running.

Once the agent is active, the adapters associated with the agent can either wait passively for events of interest to the agent, or they can actively poll the environment to see if any events of interest have occurred. In either case, when an event is detected, the adapter calls the Intelligent Agent resource manager to start the associated engine(s). In response to the event, the engine can ask an adapter (either the adapter that originated the event, or another adapter associated with the intelligent agent) to take action (sense or effect).

Internationalization: The Internationalization resource manager provides a mechanism to describe and manage the various elements of the internationalized processing environment expected by application users.

For an application to meet the specific needs and preferences of a variety of users in different countries, it needs to be internationalized at build time for general worldwide use, and it needs to be localized for a particular language, culture, and character data encoding at run time. This allows any user, at any location, to use that application and have it provide culturally correct processing results.

The description of the language, cultural, and character data encoding needs of users in a particular environment is contained in a data structure known as a locale. The Internationalization resource manager supports the definition, management, and usage of the information contained in locale data structures.

Locale source information is bound with character encoding information in a compilation process that produces the locale data structure. The data structure is also given a name as part of the compilation process. The name is provided in a recognized industry convention of a two character language identifier, followed by a two character country identifier and a character string identifying the character data encoding use. The International Organization for Standardization (ISO) standards for country names (ISO 3166) and language names (ISO 639) are used.

In addition to locale support, the following functions are provided through APIs:

- Character classification, for example common graphic versus wide characters
- Collation, based on cultural expectations
- Character string manipulation
- Formatting of date, time, and monetary character strings
- Access to multibyte or wide character data in stream I/O
- Alternate text message libraries with run time information substitution

The programming model and related interfaces of the Internationalization resource manager have been defined through the Institute of Electrical and Electronics Engineers (IEEE) Portable Operating System Interfaces (POSIX), X/Open Joint Internationalization Group (XoJIG) and UniForum (before the formation of the XoJIG) development efforts and those of the ANSI/ISO C programming language community. In addition to this work, IBM extensions are provided to the standard base support.

Virtual Machine: The Virtual Machine resource manager implements the industry standard, object-oriented Java execution environment defined by the JavaSoft division of Sun Microsystems, Inc. It provides a platform independent execution environment for Java programs such that the same binary programs will execute on all Java compliant hardware and software configurations. The primary role of the Virtual Machine resource manager is to execute Java applets that are downloaded to client systems to enhance Web browser based computing capabilities providing for example, animation or complex logic. However, the Virtual Machine resource manager can also execute Java applications on a server system. The Java object model, instruction set, and standard interfaces are the same everywhere. Developers can create applications and content with the same tools, in the same language, and using a consistent programming model on both client and server; the resulting binary programs will execute on any Java compliant virtual machine. The implementation is provided primarily by a mapping onto the other resource managers and services of an Open Blueprint distributed system.

The Virtual Machine resource manager emphasizes integrated support for networking. It supports World Wide Web technology through the use of the HTTP and Web Browser resource managers and HTML. The Virtual Machine resource manager implements a distribution paradigm where the program to be executed is generally fetched (accessed and downloaded) on demand from a server elsewhere on the network.

The Virtual Machine resource manager is a program that emulates a computer. This virtual computer has its own set of instructions which is referred to as bytecodes. It also has libraries of predefined functions for frequently used utility functions.

The main features of the Virtual Machine resource manager are:

- *An interpreter* that interprets the industry standard set of Java bytecodes. The interpreter also provides multitasking, exception handling, and the loading and execution of methods on objects. These mechanisms implement the Java object model, a dynamically linking, single inheritance model where the objects, variables, and methods are all *typed,* that is, they are identified as belonging to specific Java classes. This typing is enforced at execution time.
- A set of classes (libraries) that provide the Java defined implementations of functions including:
  - A graphical user interface - graphics and windows
  - Bytestream file input and output
  - Java Database Connectivity (JDBC) for accessing relational database data
  - A communications interface based on IP sockets
  - Collections, events, math, and other utility classes
  - A range of other functions that are currently under industry development led by JavaSoft.

The implementations of Java classes comprise mappings to other Open Blueprint resource managers and services wherever practical. For example, the graphical user interface maps to the presentation service of the host operating system such as the OS/2 Presentation Manager or the Solaris Motif implementation.

The Virtual Machine is language independent; in addition to Java, languages such as NETRexx target the Virtual Machine resource manager. However, because the Java object model is built into the Virtual Machine resource manager, and type safety is enforced at execution time, not all languages fit naturally. In particular C++ and other languages using multiple inheritance or exposing pointers are unlikely to map efficiently to the Virtual Machine resource manager.

The Virtual Machine resource manager supports both Java to Java and Java to non-Java object communication, the latter through the Object Request Broker.

The Virtual Machine resource manager will continue to evolve rapidly because the value of Java technology is based on platform independence, reduction of system administration, and being a preferred environment for innovation on the World Wide Web. These are not static attributes; they require, and JavaSoft anticipates, a sustained agenda for creating and enriching standard extension classes, and for pushing additional function into the associated classes.

Workflow: Workflow management consists of a set of functions that help to define, execute, manage, and reengineer business processes across a heterogeneous system environment. Business processes can be quite diverse, such as contracting for a purchase or processing a mortgage loan application. The Workflow resource manager is a driver of complex applications. It is a coordinating agent initiating the execution of work by people, and the execution of programs in multiple, distributed workflow-managed processes.

In many companies, valuable process definitions can be buried deep within the logic of application programs. Changing a process means changing application programs, which can be time consuming and expensive. Workflow management makes application programs serve processes by separating the process definition from the applications performing the process. It helps organizations design processes, modify them more easily, and execute them more efficiently. An organization's processes are an important asset, and companies in control of their processes have a definite competitive advantage.

The Workflow resource manager contains components to support the reengineering of business processes through process model definition and documentation tools, as well as to control and facilitate the execution of those processes in open and distributed environments.

The Workflow Manager Client Function includes support for:

- Process setup and administration
  - Process definition to capture and document work flows and test and analyze their execution before they are used
  - Administrative operations to manage workflow execution
  - Registration and configuration services
  - Post-execution analysis of logged measurement data
- Process execution
  - Work list handling including support for end user applications
  - User specified workflow client applications

The Workflow Manager Server Function includes support for:

- Management of all process models and instance state data
- Invocation and monitoring of function that is not driven by the end user
- Coordination of the state of multiple instances of multiple processes

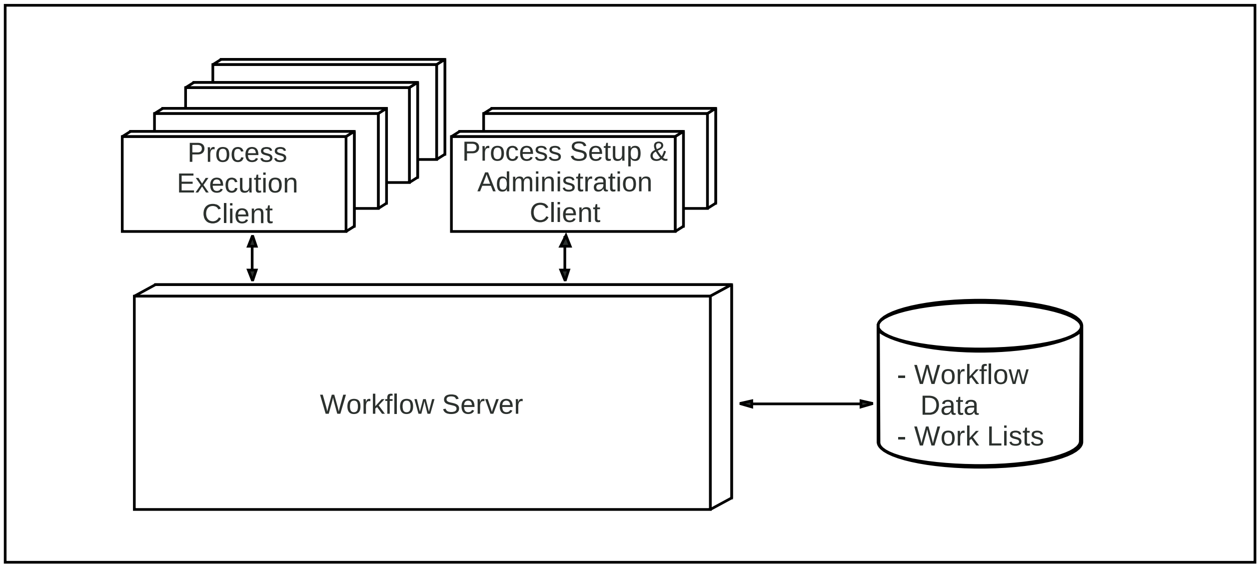

Figure 18. The Workflow Resource Manager

The Workflow resource manager interacts with and uses the services of other resource managers:

- Collaboration, to support business processes that are a mixture of collaborative, ad hoc, administrative, and production workflows
- Web Browser and HTTP, allowing World Wide Web/Internet access to workflow capabilities
- Data Access resource managers for access to files and databases
- Directory, to locate people, applications, and configurations
- Security, for identification and authentication, and access control
- Communication resource managers for all client-to-server and server-to-server communications

IBM is a member of the Workflow Management Coalition whose goal is to foster interoperability among workflow products. The coalition is addressing application programming interfaces for access to workflow management services as well as formats and protocols for communication among workflow services.

Mail: Electronic mail is supported as a mechanism for users to exchange electronic mail and as a high level service that other applications can use to send all types of materials to end users or other application programs. Applications in the distributed system can use electronic mail to transmit messages in many different forms. In addition to the interchange of unformatted text messages, the Open Blueprint Mail resource manager supports transmission of:

- Formatted documents
- Unstructured data and files
- Programs
- Graphics
- Images
- Video, audio, and other multimedia data

Mail-enabled applications will use electronic mail service for many purposes. Some examples include:

- Electronic bulletin boards, conferences, and forums
- Inter-organization business communication and transactions, including EDI
- Group scheduling and calendar systems
- Electronic publishing and distribution
- Electronic forms routing and workflow applications

A set of **open**, **industry standard APIs** provide the functional interface to the mail service for mail-enabled applications. These APIs consist of Vendor Independent Messaging (VIM), Messaging API (MAPI), and Common Messaging Call (CMC).

- **VIM**: A common client API developed by a consortium of companies, including IBM. It is intended to be a multi-platform standard that can be implemented by a variety of applications running in different environments.
- **MAPI**: An additional client API for Windows. It was developed by Microsoft and has become a de facto standard due to its pervasive use in the industry. MAPI has two flavors, simple and extended. Simple MAPI provides user applications an interface to the mail application. Extended MAPI is used for interaction between the client and the server and is one way in which mix and match clients can be achieved. The Mail resource manager supports both simple and extended MAPI.
- **CMC**: An output of the X.400 API Association (XAPIA), intended to provide access to services by an application that supports any of the existing mail-enabled interfaces (such as VIM and MAPI). CMC 1.0 is a client API, but full CMC (2.0 and later) will include interfaces to distribution lists, gateways, and other mail functions.

The pervasive and mission-critical nature of e-mail is driving increasing standardization of the mail APIs. Because CMC represents the most open approach and protects investments made in existing applications, both the Electronic Messaging Association (EMA) and XAPIA have endorsed it. Most major e-mail vendors, including IBM, Microsoft, Lotus and Novell, have announced support for CMC.

In addition to the interfaces that are used by applications to access electronic mail, there is a set of function protocols, known as electronic mail message transfer protocols, that are widely used in today's networks. The TCP/IP protocol suite provides Simple Mail Transfer Protocol (SMTP) which supports the sending of mail over the Internet through the TCP/IP SENDMAIL application. Open Systems Interconnect (OSI) standards define the X.400 Messaging Handling Services protocol. IETF defines the Post Office Protocol Version 3 (POP3). Additionally, value added network providers such as AT&T, MCI, and CompuServe support their own protocols and supply gateways to support X.400 or SMTP. Products within the Systems Network Architecture support a store-and-forward distribution system called SNA Distribution Services (SNADS). Novell NetWare provides a Message Handling Service (MHS) that is widely used in DOS systems for electronic mail. Most electronic mail vendors and network providers have recognized the need to connect to open systems protocols and at least provide gateway support from their own transfer protocols to X.400 and SMTP. The Open Blueprint Mail resource manager supports standard SMTP, X.400, POP3, and SNADS protocols. Access to other important proprietary message transfer protocols will be through gateways.

Collaboration: Collaboration is the foundation of groupware, which is software that supports teamwork and cooperation among people.

The Collaboration resource manager provides the basic functions that support collaborative activities. Collaboration relies on a:

- Shared space, which can be a virtual room, a white board, or shared on-line space
- Shared information, which is the input to and an important work product of a collaborative process

Collaboration can be between two people or it can take the form of many-to-many information sharing.

The Collaboration resource manager support organizes shared information using a paper document metaphor and places the shared information in a collaboration document store.

The document metaphor provides a model of a store of document objects where the document objects become the basis of collaborative activities. There can be multiple collaboration document stores on a client or server system. In addition, the collaboration document store logically includes structured data stored in relational databases, web sites, and public information networks.

Both synchronous and asynchronous collaboration are supported.

Synchronous collaboration is the most common type of collaboration, where activity occurs during a single period of time (although the participants can be in different locations). Examples of synchronous collaboration are audio/video conferencing and face to face meetings.

Asynchronous collaboration removes all constraints on time and location. Many copies of the collaboration document store can be spread over many locations, and the only time limit is the completion date. Asynchronous collaboration enables, for example, a document produced in the United Kingdom to be reviewed in Australia after the United Kingdom users have left work for the day. The United Kingdom users can pick up the completed reviews after they arrive at work the next morning.

The Collaboration resource manager supports sharing information that is contained in collaboration document stores. Participants can browse the information in a document store and search for a particular document or set of documents. Collaboration enables participants to edit the documents and make updates available to other participants. Alternatively, additional information or reviewers' comments can be attached to the original so the original document is preserved.

Replication makes all copies of a collaboration document store essentially identical, over time. If a user makes changes in one copy of the collaboration document store, replication ensures that those changes are added to all replicas. Replication allows collaboration document stores to be distributed to more than one location and kept synchronized. These locations could be in different time zones or different countries. The servers containing collaboration document stores connect to each other at scheduled intervals and replicate their collaboration document store documents. Server to server replication schedules are set up in the Name and Address Book (part of the federated Open Blueprint Directory) by an administrater.

The Collaboration resource manager uses the replication infrastructure to support mobile clients. Both the mail and collaboration document stores are supported. Replication enables mobile caching and reconciliation of data, that is the ability to make a replica copy of the collaboration document store on a workstation and enables mobile users to work off line on the replicated data. If changes are made to the local collaboration document stores, when the user connects again to the network and carries out the reconciliation function, changes are replicated back to the server (and updates are sent to the workstation).

Audio/video conferencing is a form of synchronous collaboration. In the Collaboration resource manager, audio/video conferencing is built on the Intel ProShare architecture base, which is rapidly becoming a standard architecture for audio/video conferencing activities. The ProShare architecture defines how conference connections are established among desktop systems. After the connections are established, individuals can participate in an audio/video conference and can share data such as files and images within a shared document store.

The Collaboration resource manager's audio/video conferencing capabilities enable users to take advantage of the collaboration document store, information access, management, and security functions to enhance audio/video conferencing. Specific conferencing facilities include maintaining conference address lists, the journaling of conference sessions, and the logging of conference activity. In addition, application program interfaces (APIs) and conference enabled messaging are supported.

The Collaboration resource manager also supports calendaring and scheduling. This support includes setting up appointments, scheduling and notification of meetings, confirmation handling, and room reservations. Additional functions include free time searches, repeating appointments and conflict warnings. In the collaborative process, this feature enables all users to participate in organizing meetings and related collaborative activities. Users can view the schedules of other users enabling them to set their own schedules accordingly.

Calendaring and scheduling are supported by the use of specialized collaboration document stores in which the documents represent, for example, conference rooms and individual schedules.

APIs and application development tools associated with the Collaboration resource manager facilitate the broadening and the customization of the collaborative environment.

While the Collaboration resource manager provides the basic collaboration support, it interfaces with other resource managers to achieve broad workgroup support. The collaborative and ad hoc workflow support in the Collaboration resource manager can be combined with the administrative and production workflow support provided by the Workflow resource manager to support a wide variety of business processes. Close association with the Mail resource managers integrates e-mail into the collaborative environment.

The Collaboration resource manager works with the Relational Database and HTTP resource managers to allow access to relational and World Wide Web data using the collaboration document metaphor.

Telephony: The Telephony resource manager enables the merger of telephone and computer technologies and provides the user with integrated voice and data services. This merger is commonly referred to as Computer Telephony Integration (CTI). The combination of verbal communication and computerized documentation is increasingly important to doing business in the modern world.

The main functions of the Telephony resource manager are **call control**, **voice processing**, and **Internet voice**.

Call Control encompasses basic client/server and direct attach telephone applications that control the simple telephone (that is, the standard 12 buttons), as well as more sophisticated call center applications that work closely with the telephone switch, providing multiple automated functions for the call center personnel.

In client/server telephony, functional requests, for example, to make a call, place a call on hold, or transfer a call, come from a source external to the connection of the phone, that is, applications that use the Telephony resource manager.

The server connects with a telephone switch, for example a Private Branch Exchange (PBX), through a CTI link. This CTI link can provide complete control over the features of a telephone without any manual interaction with the phone set. It enables the server to provide an application with information such as who is calling (automatic number identification or Caller ID), the states of the phones connected to the switch, and event progress messages indicating the actions happening with calls (on hold, conferencing, transferring). The CTI link enables an application to know as much about a call as the switch does and to follow the progress of a call even after it has been transferred from a particular telephone.

In direct attach telephony, there is no server involvement; an adapter device in the workstation permits call control through inband signalling to the telephone switch. The control link is the wire between the telephone and the switch. A subset of Telephony resource manager functions is supported in the workstation. The Telephony resource manager uses the Signalling and Control Plane to access the adapter device.

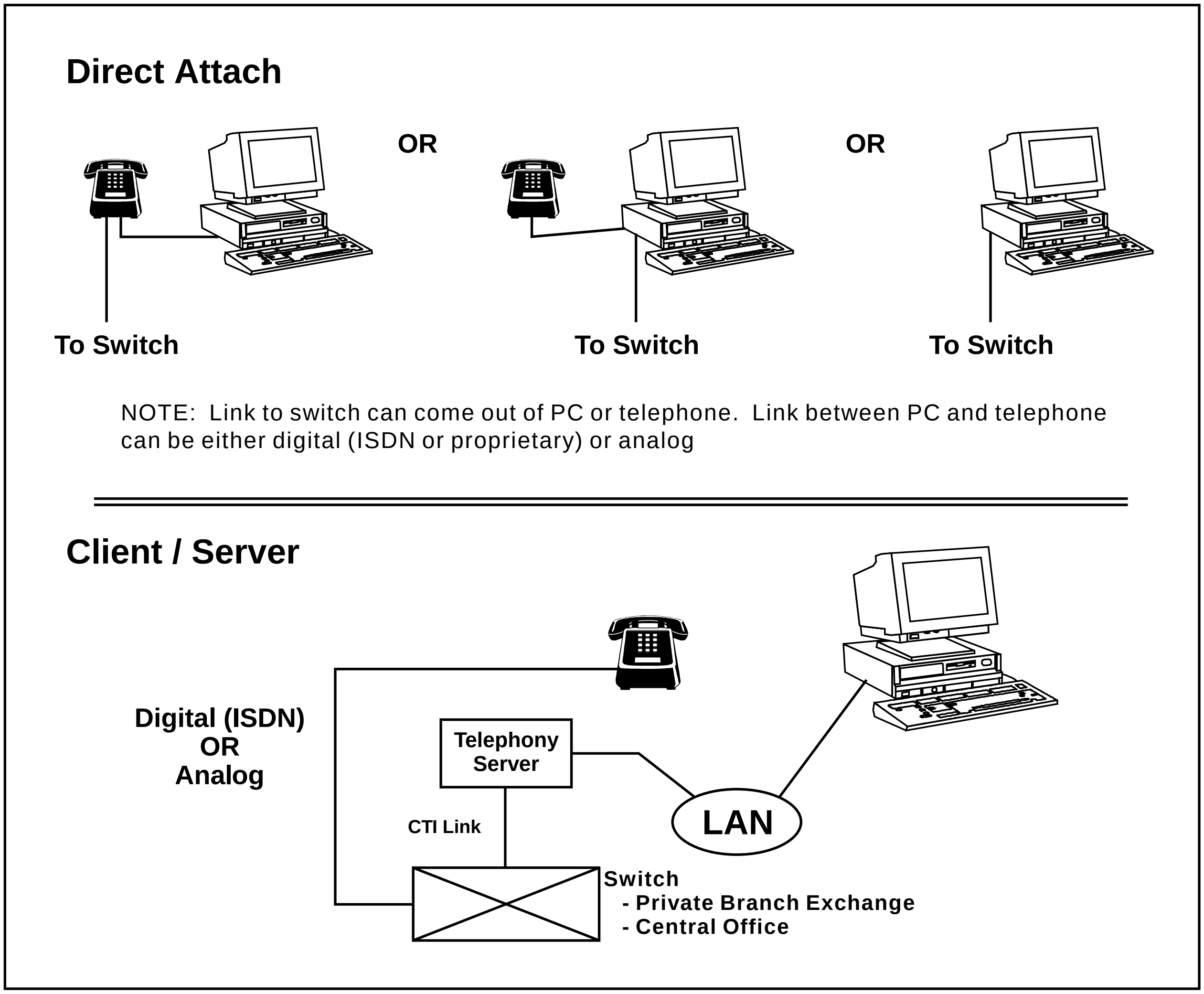

Figure 19. Direct Attach and Client/Server Telephony

The voice processing portion of the Telephony resource manager works with the digitized voice itself. It allows the telephone to be an input device to access information. Voice processing can be used to answer a call with a pre-recorded message and/or using tone recognition can begin an audio dialog with the caller, allowing the caller to request a service, such as reviewing a checking account balance. It can provide voice mail functions or place outgoing calls. Through the use of voice related technologies such as speech recognition and synthesis, it can transform voice to text and text to voice.

Internet voice allows communication over the Internet using the voice and sound features of the workstation. The voice is captured, compressed, and sent over the Internet. New enhancements to the internet protocols allow the voice packets to be prioritized, travel expediently over the network, and arrive at the destination in a timely manner that preserves the fidelity of the voice. Internet voice applications provide user interfaces and access to directory services to allow access to the desired party.

Digital Library: The Digital Library resource manager supports the storage and distribution of information objects. Libraries typically provide for information collection, organization, searching, analysis, synthesis and dissemination. The Digital Library resource manager uses and provides Open Blueprint functions to provide the end-to-end support needed to reproduce, emulate, and extend library services in an information technology environment. It manages machine-readable information objects in libraries that scale from small digital libraries to distributed libraries with petabytes[16] of data and millions of objects, both personal and enterprise confidential, along with access to commercial copyrighted materials and public free-access libraries.

Information managed includes text documents and audio, video, graphics, and image data objects. Value add to the stored information is provided by indices that enable searching via attributes and content. Separation of indices from the information allows for non-destructive, persistent annotation and organization through logical folders that can bridge across different information types and physical stores.

The Digital Library resource manager includes support for intellectual property rights management that is built on the Open Blueprint security functions for identification and authentication. Technology that protects intellectual property from misuse or misappropriation, such as watermarking, is also provided.

The Digital Library resource manager supports applications and other resource managers, such as Multimedia and Mail, by providing access to the information it manages. Underlying storage support is provided by the Data Access resource managers.

Data Access Services

Relational Database: Relational database facilities are made available through the Structured Query Language (SQL) by the Relational Database (RDB) resource manager. The ISO standard for imbedded SQL is SQL92 and the level supported by the Open Blueprint for interoperability is SQL92E (entry level). In addition to the embedded form of SQL defined by ISO SQL, the Relational Database resource manager also supports the callable interface defined by X/Open, SQL Call Level Interface (CLI), and the Open Database Connectivity (ODBC) interface defined by Microsoft for Windows.

The SQL CLI was developed to allow SQL applications to be delivered as “shrink-wrapped” applications, without including the source code. Prior to the definition of SQL CLI, customers had to process the application source code in a “pre-compile” step against their desired database. With SQL CLI, the application can be delivered as object code, or as executable binary, that automatically invokes any DBMS that supports the SQL CLI interface. ODBC extends the capability of SQL CLI to provide a routing capability, among other functions, that enables an application to call multiple, heterogeneous databases within the application by invoking the ODBC API.

The RDB resource manager is a distributed resource manager. It supports clients[17] on machines with no local database. RDB resource manager servers have a local database and interface with that database to service requests from a client. Thus, applications running on any system in a network have access to relational data residing on any other system in the network.

RDB resource manager clients use the server name specified by the application in the SQL CONNECT statement to identify the name of the desired database. This name either represents or can be mapped to the federated name of the database, which is then used to obtain information about the RDB resource manager database server from the directory. This information enables the client to establish communication with the server.

The connection to the RDB resource manager server is made using the user identity established at initial user logon (through the Open Blueprint security authentication services). Authorization for access to the tables and views is handled by the RDB resource manager server using the SQL GRANT and REVOKE statements that are part of SQL. Finally, instances of the RDB resource manager cooperate with the

[16] A petabyte is 10^{15}, or 1000000000000000.

[17] Clients are specific to an individual RDB resource manager implementation.

Transaction Manager on their local systems in order to coordinate database changes among the various RDB resource managers and with non-database resources.

Two standard RDB resource manager client-to-server protocols are defined: **International Organization for Standardization's Remote Database Access (RDA) protocol** and the more advanced **Distributed Relational Database Architecture (DRDA) protocol**. Both are function protocols. RDA flows over OSI transport protocols. DRDA uses the Open Blueprint Conversational resource manager and can flow across any transport network supported by Common Transport Semantics. The protocol supported by the Open Blueprint for interoperability is DRDA.

While many commercial databases do or will support these protocols for interoperation with other vendor databases, nearly all use proprietary protocols between their client and server components. Figure 20 shows the use of a functional protocol gateway to support heterogeneous, distributed data over any installed, supported transport.

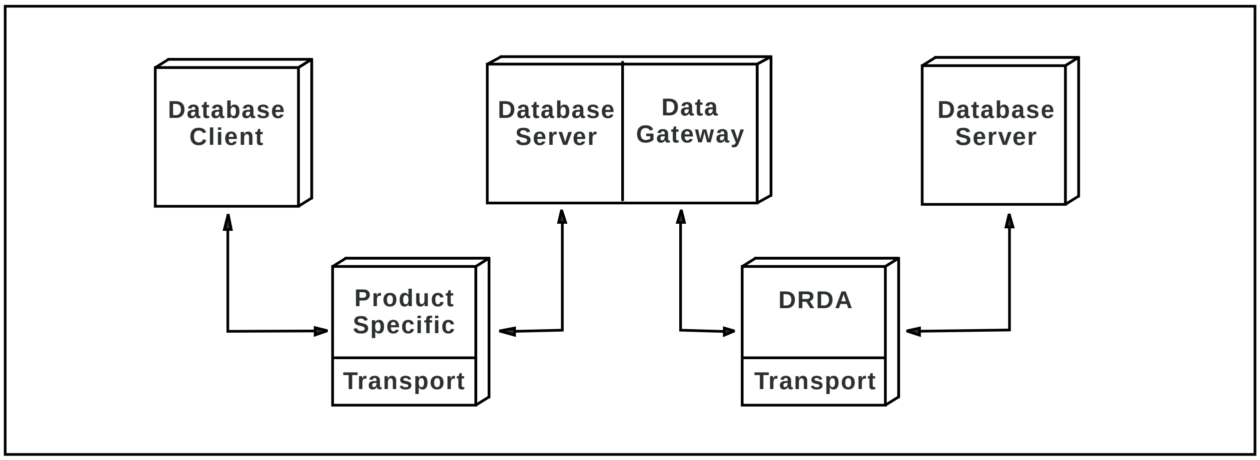

Figure 20. Non-DRDA Access to DRDA Data

The distributed RDB resource manager also provides support for data replication management for database information to optimize application performance by having the required data reside locally (or very close to) the application.

Hierarchical Database: The Hierarchical Database resource manager provides hierarchical database facilities to applications and resource managers through Data Language/I (DL/I), the de facto standard hierarchical data manipulation language. Hierarchical data is organized into a series of database records. Each record is composed of smaller groups of data called segments. Segments in turn are made up of one or more fields. Segments relate to each other as parent and child and are defined in terms of type and occurrence of segment.

Hierarchical databases are also defined in terms of type necessary to meet different application processing requirements. Data can be organized sequentially or directly and either organization may be indexed. Data Entry Databases (DEDBs), for example, provide the highest levels of availability for, and efficient storage of and access to, large volumes of detailed data.

The Hierarchical Database resource manager provides remote access to and interoperability among databases in an enterprise or network through the Open Blueprint Conversational resource manager and Network Services. Support for clients and servers (with full database management support) is provided for in a wide variety of IBM and non-IBM environments.

Object-Oriented Database: The Object-Oriented Database resource manager stores data that is not regular in structure as is, for example, record or relational table data. Computer Aided Design (CAD) data used in many engineering applications is typical of this type of data, but a growing number of other applications use non-regular data storage and access.

The Object-Oriented Database (OODB) resource manager enables the client to have transparent access to objects, regardless of location or storage format.

Transparent access to distributed objects in a client/server environment enables distributed application processing among multiple servers and multiple clients.

Applications supported by the OODB resource manager use standard C, C++, Java, and Smalltalk programming interfaces. The OODB resource manager may also be accessed through application use of the Open Blueprint Persistence Services resource manager. The client part of the OODB resource manager connects to servers on behalf of the application to access objects. The Open Blueprint Directory resource manager is used by the OODB resource manager to locate object-oriented databases in the network.

Communication between clients and servers flow across transport networks as supported by Common Transport Semantics.

The OODB resource manager uses the Access Control resource manager for authorizing object access based on the authenticated user ID established by the Identification and Authentication resource managers during user logon.

Additionally, the OODB resource manager ensures data integrity by coordinating transactions among multiple clients. Transaction management is based on serializing update and read only transactions. Transactions can be nested thus providing the capability to rollback inner transactions and still commit outer transactions. Transactions involving more than one server or involving other Open Blueprint resource managers are coordinated using the two-phase commit protocol as supported by the Open Blueprint Transaction Manager resource manager.

In addition to transaction management, data integrity is maintained with the use of archive, logging, online backup, and replication services.

Persistence Services: The Persistence Services resource manager is based on the OMG Persistent Object Service specification. It provides object-oriented applications with a common application interface that is expressed in OMG CORBA IDL, a generalized strategic syntactical interface. The language interface is provided through SOM or through specific language facilities such as C++, Smalltalk, and Java class libraries.

When application objects are to be permanently stored in an underlying data store, they are referred to as persistent objects and the services used to store and retrieve them are referred to as the persistence services. These services provide the methods for moving and mapping from the in-memory representation to the disk-resident representation of an object.

The Persistence Services resource manager offers choices to the application and object designers. A designer can make the persistence transparent. The client application can be completely ignorant of the persistence mechanism if the object designer chooses to hide it. Alternatively, the object designer can expose the persistence functions that a client may need.

The data store can be any of a wide variety of implementations including relational databases, hierarchical databases, object-oriented databases, and record files. The differences between these data stores can be hidden behind a common persistence interface.

The Persistence Services resource manager uses object designer-specified mappings when retrieving or storing objects that reside in or work with traditional data stores.

The Persistence Services resource manager allows an object-oriented application to be independent of the underlying data store. The application can be adapted to new or alternative data stores through the addition of alternative mappings.

The Persistence Services resource manager is used in conjunction with other object services such as naming, transactions, and Life Cycle. The Persistence Services resource manager, for example, does not control transactions, but it cooperates with the Transaction Manager resource manager.

File: The File resource manager includes access to multiple file systems throughout the distributed system, providing the image of a single file system. Files can be named through the Open Blueprint naming conventions. Local file system implementations will provide access to files through the distributed directory name resolution process and access control services.

File APIs can be classified as **record** and **byte stream** as follows:

Record Supports structured data (records, keys and indexes)
Byte Stream Data structures are left to application conventions

File APIs are supported by a ***file system framework*** that supports standard file APIs and routes file requests to local file system implementations or remote file systems (through one of a number of possible file clients). The file system framework insulates file applications from multiple file system implementations and the specific protocols that might be required to access a distributed file server.

The file system framework includes support for standard procedural interfaces that are commonly supported in the industry as well as emerging technology for object-oriented access to data.

File APIs are conveyed from file clients to file servers and among file servers through function protocols that are common to both the client and server environments. Typically, clients would support a limited number of file access protocols and servers would support multiple protocols. File servers would include file protocol exporters that run on top of the file system framework and translate file requests to the file APIs.

While there are a large number of file access protocols[18] that are commonly found in the industry, two file protocols are emphasized in the Open Blueprint due to the level of integration with other Open Blueprint services:

- DDM[19] (Distributed Data Management) architecture for record file access.
- Distributed File System (DFS[20]) protocols for byte stream file access. It supports a wide variety of semantics including byte stream requests, client caching, server-to-server caching, and file system administration.

The File resource manager is structured to support multiple protocols. Any file is accessible through any protocol, subject to the semantic limitations of the protocol. This is illustrated in Figure 21 on page 66. The structure allows for the possibility that the file protocol used between a client workstation and a file server is different from that used between the file server and another file server. The first file server can

[18] Examples of these are System Message Block (SMB), Network File System (NFS), and Netware Core Protocol (NCP).

[19] The DDM architecture supports several classes of file semantics including record, byte stream, and transfer. It is also used to support Distributed Relational Database Architecture DRDA.

[20] DFS is the OSF/DCE file system protocol.

thus shield the client from the multiplicity of file protocols, as well as possibly provide local caching of remote files, or exploitation of a larger file server for archiving or back up of files.

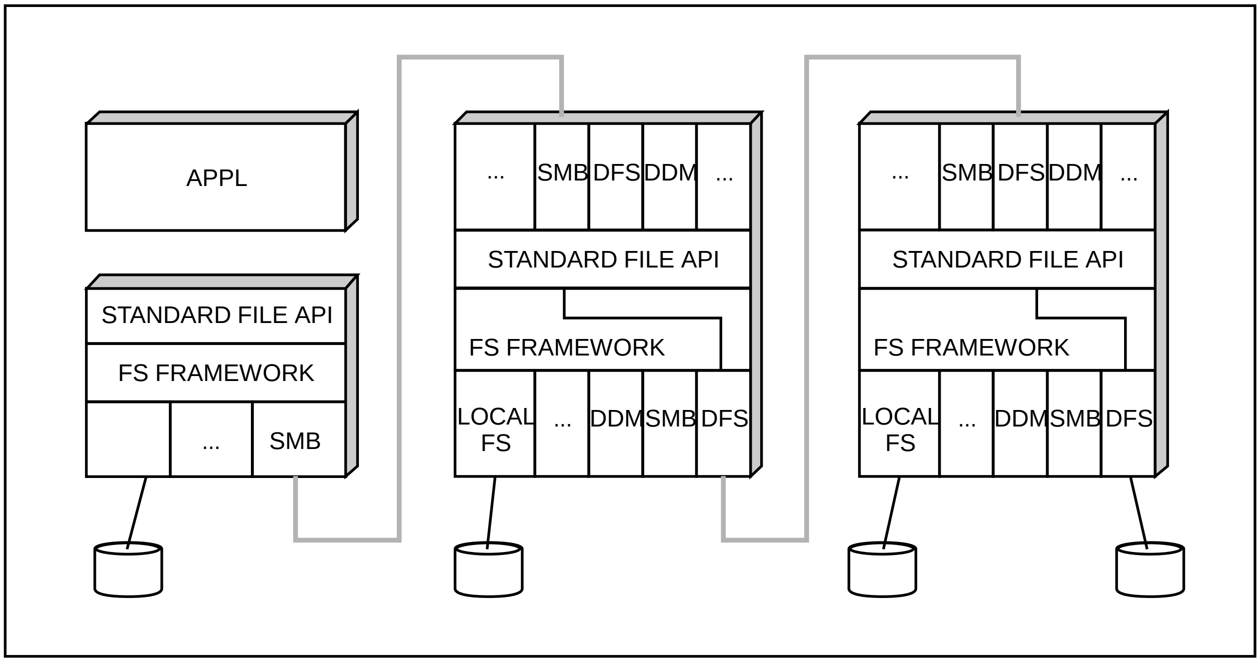

Figure 21. Support for the Heterogeneous File Environment

File resource manager implementations integrate with the Open Blueprint Identification and Authentication resource manager to exploit the user logon context established at the client (not requiring separate logon or user enrollment). Access to specific files can be controlled using the Open Blueprint Access Control resource manager. File protocols support transmission over any standard networking transport by virtue of using transport independent communication resource managers. When file gateways are involved, the Open Blueprint naming, location independence, and single signon objectives are maintained on an end-to-end basis (through the gateway). The File resource manager interacts with the Transaction Manager to support transaction processing.

Storage Management: The Storage Management resource manager provides protection and cost effective management of storage resources. These facilities are provided in both a transparent service and an API service.

Storage management consists of the following four elements:

Management Class Defines storage availability, space, and retention attributes

Data Class Defines the actual definition of the data

Storage Class Defines performance and device availability requirements

Storage Group Defines the set of storage volumes for future allocation

The transparent services of storage management integrate primarily with the File resource manager. If applications use the base file systems, then the storage management constructs will be automatically utilized. These include services such as:

- File protection through state-of-the-art backup systems
- Long-term file or named binary object preservation through data archiving

- Infinite storage constructs where infrequently accessed data is migrated to more cost effective storage components. On access, the data is transparently recalled to the file system for standard access.

All of these services operate transparently to the user or application and thus provide for data protection and management in a seamless manner. They all apply the storage policies as defined in the above storage management classes.

The requirements of some application areas/resource managers such as Database and Digital Library are not always met by file systems. For these, the Storage Management resource manager provides the ability to directly manage the archiving and backup of data without their having to go through the file systems. The API provided is based on the emerging X-Open specification entitled *Systems Management: Backup Services: API (XBSA)*.

The Storage Manager integrates with the Open Blueprint security services to insure protection of the storage resources. It also integrates with transport services to provide for distributed management of the storage resources within an enterprise. Other components of the Open Blueprint that storage management uses include Transaction Manager and Directory.

Application Development

The Open Blueprint structure defines an application as the use of information technology to assist in the execution of a business process (see “Application Focus” on page 13). Distributed, network applications execute in or exploit multiple systems to accomplish their functions.

The Open Blueprint structure provides the reference model for services which enable applications to use system components and infrastructure across a distributed environment. The Open Blueprint resource managers provide common run time services and facilities which deliver consistent and interoperable functions across an application's network topology. The richness of these resource managers' functions allows the developer to easily incorporate network capability into the application. Application development tools enable the exploitation of these facilities to deliver integrated, interoperable business solutions. During development, tools create application elements that access Open Blueprint resource managers through calls on the various resource manager Application Programming Interfaces (APIs). Since the APIs are based on standards, the application can be delivered into many different computing environments and can be protected from change in the underlying run time services and facilities. These services and facilities are available through multiple programming facilities and languages, both procedural and object-oriented.

The Open Blueprint structure defines the environment for distributed and client/server applications. This environment is heterogeneous and may include multiple operating systems and instances of resource managers, such as relational databases and transaction monitors, supported by multiple transport network protocols. From the development perspective, the expectation is that tools will provide a balance between exploiting and masking these differing facilities such that applications can execute within the environment. In doing so, the tools should allow the developer to focus on the business problem rather than the underlying technology and infrastructure of the execution environment.

Distribution Models: Distributed applications range from simple queries to robust, complex, distributed solutions.

Simple client/server applications allow the customer to gain benefits from implementing graphical user interfaces (GUIs) and by providing access to remote data. With an object-oriented programming model, these interfaces are provided through visual parts, adding additional portability and transparency to the application and improving development productivity through reuse.

Distributed logic applications are enabled through a number of Open Blueprint resource managers including the Relational Database resource manager (stored procedures), the Communications Services resource managers (Remote Procedure Call, Conversational, Messaging and Queuing, and HTTP), and the CORBA-compliant Object Request Broker.

Distributed solutions employ both distributed data and distributed logic across multiple locations. The Open Blueprint structure enables these solutions with interfaces which provide transparency and interoperability across various application topologies. Distributed business objects provide further flexibility and location transparency to object-oriented applications which are built based on Open Blueprint Object Management Services. This allows the application developer to focus on building the business objects and placing them where it makes sense while still maintaining access across the distributed topology.

The distributed applications built from the models just described may be combined into larger business solutions using workflow functions.

Development for Distribution: Application Development in the Open Blueprint supports four types of development environments: Third-Generation Language Development, Fourth-Generation Language Development, Object-Oriented Language Development, and Scripting Languages. The combination of these development environments with the Open Blueprint resource managers leads to productive and effective solutions for developers. For example, for many data query and update applications, a screen builder and a 4GL programming language simplify the application development process. Also, scripting languages, when combined with the parts capabilities of the Open Blueprint Compound Document resource manager, may provide significant benefit in bringing business objects together at the desktop.

Complex, mission critical applications are still largely custom designs. With increasing frequency, the design process used to develop such applications is one of iteration and successive prototype development, instead of the one pass, waterfall approach. In this approach, selected elements of the application are successively prototyped. Each prototype is more functionally complete than the previous one. Tools that support the iterative prototyping process include visual builders for client interfaces in IBM's VisualAge, and rapid application development and test capabilities of IBM's VisualAge Generator which allow both client and server logic to be exercised in the development environment.

Additional development paradigms, appropriate to object manipulation and usage between objects, are emerging. Frameworks, or collections of classes, are designed to work together to generally define an architecture for a part of an application. These frameworks provide the base classes which allow a developer to create and implement subclasses to complete the function needed for specific application implementations. Frameworks provide an overall structure for applications, while allowing for flexibility in how the applications can be distributed.

Tool Support and Directions: The structure of the Open Blueprint increases the ability to provide tools which support distributed needs. Tools provide the application developer with the capability of building business applications that can easily exploit any of the services provided by Open Blueprint resource managers. And, using Open Blueprint distributed services, development tools can provide the ability to transparently test distributed applications across multiple platforms.

Developing a distributed application is for the most part the same as developing a non-distributed, or stand alone, application. In the development of a stand alone application, the developer has to contend with interfaces to the user, interfaces to system resources, use of data, and implementing the business logic.

Developing a distributed application, however, requires that the application be divided into interacting elements that may execute on different systems. Each element can be developed as a non-distributed application, that is, with consideration for user interface, system resource interface, data, and business logic as appropriate. The new consideration added as a result of the application being distributed is the interaction between the application elements.

Tool support can assist in the following:

- Approaching the development of a distributed application by recording and analyzing requirements, modeling the supported business process, and visualizing the distributed solution
- Designing a distributed application by helping identify interacting application elements and analyzing the interactions among them
- Implementing a distributed application through generating application element skeletons from the design and allowing the developer to complete logic specifications, then building the elements
- Evaluating a distributed application through distributed debuggers and performance recording and simulation
- Deploying a distributed application through enabling the tailoring of elements and their distribution to target execution platforms

Applications for the various distribution topologies can be built with a wide number of tools and tool suites. In selecting development tools, current tools and current skills of the developers often weigh heavily in the decision.

In development environments such as the four described in "Development for Distribution" on page 68, there are no clear demarcations which force complete dependence on any specific level of abstraction. Within any one of these, the developer can choose to write at a lower level. However, the direction for application development tools is to focus less and less on the lower level technology and more on productivity improvements which sharpen the focus on solving the business problem. With visual building interfaces, assistants, and pre-built components, applications can easily exploit existing and shared functions and objects.

IBM provides a rich set of tool suites for developing distributed applications. These suites contain tools that are themselves distributed applications which use the Open Blueprint resource managers during development time. They share common facilities which support the concepts of team programming and enhance both individual and team development. The major suites are built around 3GLs, 4GLs, and object-oriented languages.

All of these tool suites construct application elements which interface with Open Blueprint resource managers through APIs or visual parts. The infrastructure provided by the Open Blueprint resource managers beneath these APIs and parts enables distribution, portability, and interoperability for the applications produced by the tools.

Systems Management

Manageability of hardware and software resources is a critical requirement in the open, distributed environment. For example:

- Software must be able to be distributed electronically and installed either locally or remotely.
- Configuration information for resources should be self-defining and minimize user involvement.
- Basic error logging and fault isolation functions are required to support manual and automated problem determination and resolution.
- Common standard management definitions and protocols should provide for the collection of error and performance data and the application of changes and fixes.
- Management facilities for systems administrators are required for both local and remote resources. Installations should be able to choose either a centralized or distributed management scheme.
- Standard APIs for management are needed to facilitate interoperability and application portability.

The Open Blueprint structure includes a backplane that allows for resources to be managed across multi-vendor, open, enterprise-wide distributed systems. This is enabled through the systems management component structure. The structure is based on Tivoli Systems TME 10. TME 10 defines the necessary management applications, management services, and structure. TME 10 and the Open Blueprint describe complementary aspects of capabilities needed to address a multi-vendor enterprise environment.

Systems Management Component Structure

Resource managers, as described by the Open Blueprint, will take advantage of a common set of management services. The required common management services are those prescribed by TME 10. In the Open Blueprint, systems management is shown as a backplane to emphasize its importance across all resource managers and to indicate that additional system management services are needed. Figure 22 on page 71 shows the structure of this backplane. As depicted, this structure consists of four major elements.

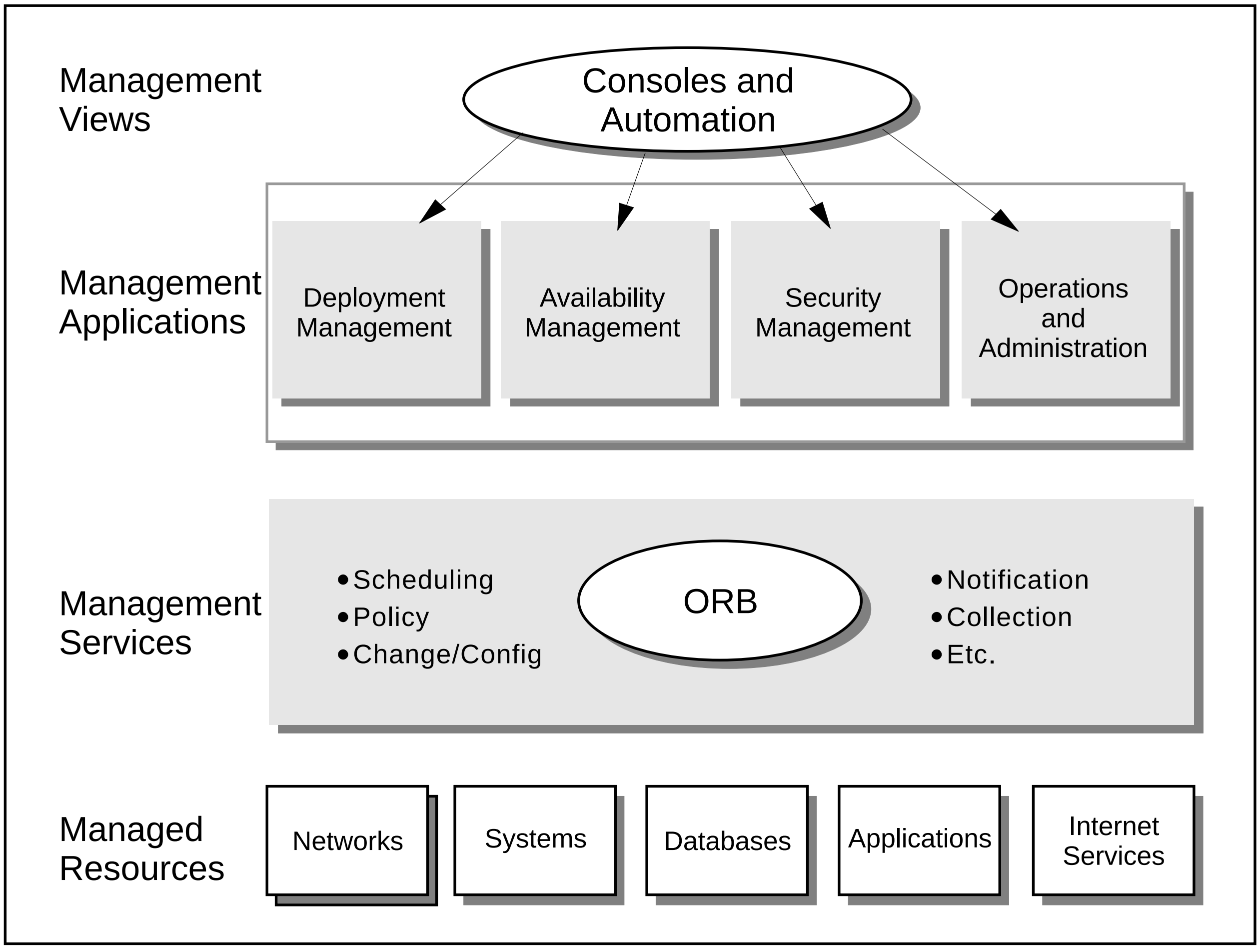

Figure 22. Systems Management Component Structure

Systems Management Views: Provide common access to systems management end user data. Views provide a common way to see and handle sets of systems management data.

Systems Management Applications: Provide functions to interact directly with systems administrators and automate management tasks. Management applications are categorized into four disciplines.

- Deployment Management

  Supports tasks associated with the configuration and change management activities for all of the frequently changing components of the enterprise.

- Availability Management

  Gathers, collects, and routes information regarding the state of all aspects of the enterprise computing environment for proactive management action.

- Security Management

  Ensures that users have access to the applications and data they need to do their job. Security management applications provide administrative support for Open Blueprint Security resource managers.

- Operations and Administration

Facilitates the automation of activities that provide the integrity and reliability of the entire computing environment. Operations and administration applications also support the administration of resource managers and their resources.

Systems Management Services: Provide common services for resource managers and System Management Applications unique to the needs of systems management. Through the use of these services Systems Management Applications can have an integrated appearance, share data between applications, and be shielded from the underlying technology used to access management information. The Systems Management Services can be placed in three categories:

- Framework Services

  Provide basic object specification, creation, interaction, and method invocation. Framework services are provided in the implementation of the CORBA compliant Object Request Broker (ORB) and Basic Object Adapter (BOA).

- Application Services

  Provide the fundamental services for modeling or storing objects for management purposes. These services define the set of intrinsic operations that all policy driven objects can inherit and implement.

- User Interface Services

  Offer a set of platform and presentation independent user interface services that allow simplification of application development.

Managed Resources: Are resources owned by resource managers. They are available for interrogation and control by Systems Management Applications.

Agent Services are used by the resource managers to access and set the state of their managed resources. Agent services insulate resource managers from awareness of the location of the managing system. They include the ability to:

- Send notifications
- Collect vital product data
- Respond to operational commands and requests for state information
- Collect performance information
- Enforce software license policies
- Capture error and problem determination information
- Request distribution of software

Systems Management Enablement of Applications and Resource Managers

The TME 10 Application Management Specification (AMS) is the foundation for systems management enablement of resource managers and distributed applications. To effectively manage an application, the management application needs information on how it is structured and how to control and monitor it. The AMS provides a format and content for Application Description Files (ADFs) which allow developers of these applications to describe standard management information.

An Application Description File contains the following:

- **Application Topology**, a specification of how the distributed application is broken up into components and how these components relate to each other

- **Application Distribution**, a list of source files and directories that make up the application

- **Application Installation**, identification of scripts to run before or after application distribution

- **Dependency Checking**, identification of scripts to check the hardware and software configuration on target systems to ensure that application requirements are met
- **Application Monitoring**, specification of what metrics and events can be retrieved from the application, and how to obtain them
- **Operational Control**, identification of scripts to perform an arbitrary set of operational tasks

Systems Management Standards

The Open Blueprint systems management supports the following industry standards:

- IETF

  TME 10 supports the Simple Network Management Protocol (SNMP).

- Object Management Group

  The Common Object Request Broker Architecture (CORBA) allows interoperability among independently developed object-oriented applications across heterogeneous computer networks.

- X/OPEN and OMG

  The Systems management services in TME 10 incorporate the Common Management Facilities (XCMF) standardized by X/OPEN and OMG. These facilities include a naming service, event service, and life cycle service.

- Desktop Management Task Force (DMTF)

  The Application Management Specification is based on the software Management Information File (MIF) as defined by DMTF.

- POSIX

  User administration is compliant with the POSIX 1387.3 definition for user groups.

In addition, there are emerging standards being incorporated into Open Blueprint Systems Management:

- IETF SNMP Version 2
- Internet standards

Local Operating System Services

The local operating system services are local resource managers and services that support the Open Blueprint's distributed resource managers. Local resource managers manage local system resources such as CPUs or devices. Local operating systems services include:

- Work management

  The local operating system provides services for the initiation, management, and termination of work requests and applications. Distributed resource managers or servers participate in the work management scope. For example, mechanisms must exist to inform a server when a client terminates. This function is typically provided by a Communication resource manager, or an indication is carried in the formats and protocols of higher level resource managers.

- Environment state support

  Facilities for storage and access to information that must be held such that activities can be coordinated across the distributed environment. Examples of environment state information are the authenticated user ID of the user on whose behalf a request is being made, and the unit of work ID that identifies a recoverable transaction. In the first example, the receiving system's communication resource manager will store the user ID such that it can be used locally for access control purposes; in the second example, the unit of work ID is stored so it can be used as coordination data for distributed transaction management. The usage of this support is by local instances of distributed resource managers, primarily the communication resource managers.

- Event handling

  Facilities are required for triggering and scheduling work when a specific event occurs.

- Local system logon

  Facilities are required to do local authentication and bridge to network authentication before any network-based activity can be started.

- Security context management

  Facilities for the storage of and access to security credentials are needed in all local operating systems. A security context must be established at the server end of a remote work request so that access control functions may be performed by the server in validating the legitimacy of the work request. Typically, the authenticated user ID is delivered to individual servers which must accept the responsibility for access control over the resources they manage. The servers may also use the local operating system's access control functions for access validation.

- Multimedia system services

  Facilities for device management and data stream control/synchronization to support physical multimedia devices such as cameras, speakers, and displays. It is necessary to provide facilities so that adequate system resources can be made available to time critical applications, resource managers, and devices supported by them.

- Locking Service

  A common locking service is required for serializing access to resources. Examples of resources that require serialization are programs, data, and devices, but the locking service does not require knowledge of the type of resource its locks represent.

  Deadlock detection and avoidance are important considerations for resource managers and applications. In the distributed environment, the problem of detecting deadlocks is increased significantly due to the increase in the number of resources involved and the potential for increased hold time across communication links. The complexity and overhead costs of detecting deadlocks increases dramatically also. The current technology uses timers to detect and terminate processes that fail to complete within the upper limit of the expected time.

- Accounting

  Facilities for collecting resource accounting data as appropriate for the system.

- Tracing

  Facilities for tracing system events and for passing on requests to active resource managers to perform tracing of events meaningful in the system.

- Journaling

  Facilities to provide a journal of events such as job completions, error records, and audit trails.

- Program Management

  Facilities including linkers and program loaders which together provide dynamic link libraries are required to be provided in a relatively compatible manner so as to facilitate application porting and reuse across a wide range of platforms. Dynamic link libraries support object class libraries.

- Runtime Environment/POSIX

  Either language runtime support or Language Environment products provide standardized services for storage allocation and management, message generation, exception handling, time and date services, and a set of mathematical subroutines. These services are available to application programs written in programming languages through a standard set of CALLs.

  The common denominator of base operating system support for applications is provided on a number of platforms by the facilities described by programming interfaces specified through the POSIX initiatives, many of which have been adopted as X/Open standards. That is, local operating system resource managers will adhere to the POSIX, or X/Open, suite of standards and standards proposals for common operating system services and interfaces to those services, such that portability of the components and applications using those interfaces and services is possible, and can be facilitated.

  Some of these components and applications are, in fact, other technology initiatives of the Open Blueprint.

Appendix A. Standards

The following table lists some principal industry interfaces and protocols in the Open Blueprint structure. The Open Blueprint structure is based on formal standards that have achieved widespread industry acceptance. The Open Blueprint structure also includes several *de facto* standards, either in the absence of a formal standard, or to achieve interoperability with existing installations.

The references for ISO/IEC, IETF, OMG, and ITU-T standards are to standards number. The references for X/Open are to an International Standard Book Number (ISBN). The references for company documented interfaces and protocols are denoted by an asterisk (*). You can get information on these interfaces and protocols from the contributing company. The IBM documents are listed below. For convenience, URLs are provided to enable access to the various organizations' Web pages.

Although X/Open and OSF have combined to form The Open Group, the X/Open and OSF references are used.

IBM Interfaces	Document Order Number	IBM Protocols	Document Order Number
CICS API	SC33-1007	APPC	SC31-6808
DL/I	SC26-8015	APPN	SC30-3422
IMS TM API	SC26-8017	DDM	SC21-9526
MQI	SC33-0850	DRDA	SC26-4651
OS/2 PM API	G25H-7103/4	IPDS	S544-3417
		MPTN	GC31-7073/4
		NetBIOS	SC30-3587
		SNA Sync Point	SC31-8134
		SNA Formats	GA27-3136

The referenced standards documents can be located and ordered at the following web sites:

Organization	URL
The ATM Forum	http://www.atmforum.com
Component Integration Laboratories (CILabs)	http://www.cil.org
The Desktop Management Task Force (DMTF)	http://www.dmtf.org
ECMA	http://www.ecma.ch
International Electrotechnical Commission (IEC)	http://www.iec.ch
The Institute of Electrical and Electronic Engineers, Inc. (IEEE)	http://www.ieee.org
International Organization for Standardization (ISO)	http://www.iso.ch
International Telecommunication Union - Telecommunication (ITU-T)	http://www.itu.ch
Internet Engineering Task Force (IETF)	http://www.ietf.org
Open Software Foundation (OSF)	http://www.osf.org
Object Management Group (OMG)	http://www.omg.org
National Center for Supercomputing Applications (NCSA)	http://www.ncsa.uiuc.edu
Worldwide Web Consortium (W3C)	http://www.w3.org
X/Open Company Ltd (X/Open)	http://www.xopen.org

Network Services

Common Transport Semantics

Interfaces	Protocols	Organization	Reference
	XMPTN	X/Open	ISBN 1-859120-42-3
	OSI over TCP/IP	IETF	1006
	NetBIOS over TCP/IP	IETF	1001/1002

Transport Services

Interfaces	Protocols	Organization	Reference
XTI		X/Open	ISBN 1-85912-049-0
Berkeley Sockets		Berkeley	4.3 BSD Document
Winsock		Microsoft	*
	TCP/IP	IETF	791, 793, 768, 1889
	IPSec	IETF	1825-1829
	SSL	IETF	draft-freier-ssl-version3-01
	RSVP	IETF	draft-ietf-rsvp-spec-13
	APPN	APPN Implementors Workshop	IBM - *
	NetBIOS	IBM	*
	IPX	Novell	*
	Appletalk	Apple	*

Signalling and Control Plane

Interfaces	Protocols	Organization	Reference
	ISDN	ITU-T	Q.931, Q.2931
	Conferencing	ITU-T	H.320

Subnetworking

Interfaces	Protocols	Organization	Reference
	LAN (Ethernet, Token Ring)	ISO/IEC	8802
	ISO DLC	ISO/IEC	3309, 4335, 7776
	Frame Relay	ITU-T	I.233
	ATM	ITU-T	I.150, I.327, I.361/2/3
		ATM Forum	UNI 4.0

Distributed Systems Services

Conversational

Interfaces	Protocols	Organization	Reference
CPI-C		X/Open	ISBN 1-85912-135-7
	APPC	IBM	*
	OSI TP	ISO/IEC	10026

RPC

Interfaces	Protocols	Organization	Reference
DCE RPC	DCE RPC	X/Open	ISBN 1-85912-041-5

Messaging and Queuing

Interfaces	Protocols	Organization	Reference
MQI		IBM	*

HTTP

Interfaces	Protocols	Organization	Reference
CGI 1.1		NCSA	http://hoohoo.ncsa.uiuc.edu/ cgi/interface.html
	HTTP 1.0	IETF	1945
	HTTP 1.1	IETF	draft-ietf-http-v11-spec-07
	HTTPS	Netscape	*
	SHTTP	IETF	draft-ietf-wts-shttp-03

Object Management

Interfaces	Protocols	Organization	Reference
CORBA 1.1		OMG	CORBA 1.1 (12/91)
CORBA 2.0		OMG	CORBA 2.0 (07/95)
CORBAservices		OMG	CORBAservices
Collections			
Externalization			
Identity			
Life Cycle			
	IIOP, DCE-CIOP	OMG	CORBA 2.0 (07/95)

Directory

Interfaces	Protocols	Organization	Reference
XDS		X/Open	ISBN 1-85912-007-5
XOM		X/Open	ISBN 1-85912-008-3
Federated Naming		X/Open	ISBN 1-85912-045-8
LDAP		IETF	1823
Naming Service		OMG	CORBAservices
	Cell Directory Service	X/Open	ISBN 1-85912-078-4
	X.500	ISO/IEC	9594 (88 Standard)
	Domain Name Service	IETF	1034,1035
	LDAP	IETF	1777
	URL	IETF	1738

Security - Identification and Authentication

Interfaces	Protocols	Organization	Reference
DCE SEC_RGY	DCE Registry	OSF	DCE 1.2
GSSAPI		IETF	1508, 1509
Security Service		OMG	CORBAservices
	Kerberos	MIT/OSF	Kerberos 5 Beta 7

Security - Access Control

Interfaces	Protocols	Organization	Reference
DCE SEC_ACL	DCE ACLs	OSF	DCE 1.2 and POSIX 1003.1e, 1003.2c

Security - Certificate Management

Interfaces	Protocols	Organization	Reference
	Certificate Definition	ITU-T	X.509v3

Time

Interfaces	Protocols	Organization	Reference
DCE DTS	DCE DTS	X/Open	ISBN 1-85912-067-9
Time Service		OMG	CORBAservices

Transaction Manager

Interfaces	Protocols	Organization	Reference
Transactional C	Encina	Transarc	*
TX		X/Open	ISBN 1-85912-094-6
XA		X/Open	ISBN 1-872630-24-3
CICS API		IBM	*
IMS TM API		IBM	*
	SNA Sync Point	IBM	*
Object Transaction Service		OMG	CORBAservices

Application Enabling Services

Human Computer Interaction

Interfaces	Protocols	Organization	Reference
OS/2 PM		IBM	*
Motif		OSF	MOTIF Specification
Windows		Microsoft	*
OpenGL		Silicon Graphics	*
X Window Xlib		X/Open	ISBN 1-872630-11-1
X Window Xt intrinsics		X/Open	ISBN 1-872630-14-6
	X Window System Protocols	X/Open	ISBN 1-872630-13-8

Print

Interfaces	Protocols	Organization	Reference
Distributed Print API		IEEE	POSIX 1387.4/D8
	DPA	ISO/IEC	10175-2
	lpr/lpd	IETF	1179
	NCP	Novell	*

Print Data Streams

Interfaces	Protocols	Organization	Reference
	IPDS	IBM	*
	PCL	HP	*
	PostScript	Adobe Systems	*

Web Browser

Interfaces	Protocols	Organization	Reference
	HTML 2.0	IETF	1866
	HTML 3.2	W3C	http://www.w3.org/pub/WWW/TR/WD-html32

Relational Database

Interfaces	Protocols	Organization	Reference
SQL (1992E)		ISO/IEC	9075
SQL CLI		X/Open	ISBN 1-85912-081-4
ODBC		Microsoft	*
	DRDA	IBM	*

Hierarchical Database

Interfaces	Protocols	Organization	Reference
DL/I		IBM	*

Persistence Services

Interfaces	Protocols	Organization	Reference
Persistent Object Service		OMG	CORBAservices

File

Interfaces	Protocols	Organization	Reference
	Distributed File System	OSF	DFS Specification
	SMB	X/Open	ISBN 1-872630-45-6
	NFS	SUN	*
	NCP	Novell	*
	DDM	IBM	*

Storage Management

Interfaces	Protocols	Organization	Reference
Backup Services (XBSA)		X/Open	ISBN 1-85912-056-3

Transaction Monitor

Interfaces	Protocols	Organization	Reference
CICS API		IBM	*
IMS TM API		IBM	*

Event Services

Interfaces	Protocols	Organization	Reference
Event Notification Service		OMG	CORBAservices

Internationalization

Interfaces	Protocols	Organization	Reference
Locale Services		X/Open	ISBN 1-85912-002-4

Compound Document

Interfaces	Protocols	Organization	Reference
OpenDoc		CILabs	*

Virtual Machine

Interfaces	Protocols	Organization	Reference
Virtual Machine Invocation		JavaSoft	*
Native Method Invocation		JavaSoft	*
Remote Method Invocation		JavaSoft	*
Debugging Tool Interfaces		JavaSoft	*
JIT Compilation Hooks		JavaSoft	*

Mail

Interfaces	Protocols	Organization	Reference
Vendor-Independent Messaging (VIM)		VIM Consortium	*
Messaging API (MAPI)		Microsoft	*
Common Messaging Call API		XAPIA	*
	SMTP	IETF	821,822
	X.400 Messaging Handling Services	ISO/IEC	10021
	POP3	IETF	1938
	SNADS	IBM	*
	MIME	IETF	1521,1522

Telephony

Interfaces	Protocols	Organization	Reference
Computer Telephony Integration (CTI)	CSTA	ECMA	179, 180

Systems Management

Interfaces	Protocols	Organization	Reference
XMP		X/Open	ISBN 1-872630-32-4
SNMP	SNMP	IETF	1157
SNMPv2	SNMPv2	IETF	1907
XCMF		X/Open	ISBN 1-85912-047-4
User Admin.		IEEE	POSIX 1387.3
	MIF	DMTF	*
	AMS	Tivoli	*

Local Operating System Services

Interfaces	Protocols	Organization	Reference
System calls, libraries		ISO/IEC	9945-1
Commands and utilities		ISO/IEC	9945-2
Threads		IEEE	POSIX 1003.1c

* The contributing companies may be contacted for more details. The IBM document numbers are on page 77.

Bibliography

Open Blueprint Publications

Introduction to the Open Blueprint, G326-0395

Open Blueprint Technical Reference Library, SBOF-8702 (printed), SK2T-2478 (CDROM). This publication includes this *Technical Overview*, the *Introduction to the Open Blueprint* listed above, and component description papers for each Open Blueprint component. Each of the component description papers is also orderable individually in hard copy. The list below provides ordering information.

Application Development in the Open Blueprint	GC23-3909
Asynchronous Transfer Mode and the Signalling and Control Plane in the Open Blueprint	GC23-3910
Open Blueprint Collaboration Resource Manager	GC23-3912
Open Blueprint Compound Document Resource Manager	GC23-3913
Open Blueprint Conversational Resource Manager	GC23-3914
Open Blueprint Digital Library Resource Manager	GN62-0039
Open Blueprint Directory Resource Manager	GC23-3915
Open Blueprint Event Services Resource Manager	GC23-3916
Open Blueprint File Resource Manager	GC23-3917
Open Blueprint Hierarchical Database Resource Manager	GC23-3918
Open Blueprint HTTP Resource Manager	G325-6593
Open Blueprint Human Computer Interaction Resource Manager	GC23-3933
Open Blueprint Intelligent Agent Resource Manager	G325-6592
Open Blueprint Internationalization Resource Manager	G325-6583
Open Blueprint Mail Resource Manager	GC23-3919
Open Blueprint Messaging and Queuing Resource Manager	GC23-3920
Open Blueprint Multimedia Resource Manager	GC23-3921
Open Blueprint Network Services	GC23-3932
Open Blueprint Object Oriented Database Resource Manager	GC32-3922
Open Blueprint Object Broker Resource Manager	GC28-1213
Open Blueprint Persistence Services Resource Manager	GC23-3923
Open Blueprint Print Resource Manager	GC23-3924
Open Blueprint Relational Database Resource Manager	GC23-3925
Open Blueprint Remote Procedure Call Resource Manager	GC23-3926
Open Blueprint Storage Management Resource Manager	GC23-3927
Open Blueprint Telephony Resource Manager	GC23-3928
Open Blueprint Time Resource Manager	GC23-3929
Open Blueprint Transaction Manager Resource Manager	GC23-3930
Open Blueprint Transaction Monitor Resource Manager	GC23-3931
Open Blueprint View Resource Manager	GC23-3934
Open Blueprint Virtual Machine Resource Manager	GC25-6588
Open Blueprint Web Browser Resource Manager	G325-6589
Open Blueprint Workflow Resource Manager	GC23-3935
Security in the Open Blueprint	GC23-3936
Systems Management in the Open Blueprint	GC23-3937

Open Blueprint Support For Mobile Computing is available in softcopy at http://www.software.ibm.com/openblue/id1u2/cover.htm

The Open Blueprint home page, located at http://www.software.ibm.com/openblue, contains information related to the Open Blueprint. It is dynamic, with new information being added as it is available.

Other Publications

X/Open Architecture Framework, available through The Open Group at their Web site, http://www.xopen.org/public/arch

Open Systems and IBM: Integration and Convergence, Gray, London, McGraw-Hill, 1993.

Essential Client/Server Survival Guide, Robert Orfali, Dan Harkey, and Jeri Edwards, Van Nostrand Reinhold, 1994

The Essential Distributed Objects Survival Guide, Robert Orfali, Dan Harkey, and Jeri Edwards, John Wiley & Sons, Inc., 1996.

Groupware: Communication, Collaboration, and Coordination, Lotus Development Corporation, 1995. Part Number 17998, also available at their Web site http://www.lotus.com

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

GC23-3808-02