Open Blueprint

GC23-3808-01

Technical Overview

Open Blueprint

GC23-3808-01

Technical Overview

Note!

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

Second Edition (June 1995)

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 55JA, Mail Station P384
522 South Road
Poughkeepsie, NY 12601-5400
United States of America

FAX: (International Access Code)+1+914+432+9405

IBMLink (United States customers only): KGNVMC(D58PUBS)
IBM Mail Exchange: USIB2NZL at IBMMAIL
Internet: d58pubs@vnet.ibm.com

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. Any functionally equivalent product, program, or service that does not infringe any of IBM's intellectual property rights 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 and Service Marks

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

9037 Sysplex Timer
Advanced Function Presentation
Advanced Function Printing
AFP
APPN
CICS
Common User Access
CUA
Distributed Relational Database Architecture
DRDA
ES/9000
IBM
Information Warehouse
Intelligent Printer Data Stream
IPDS
Language Environment
OS/2
SystemView
Workplace Shell
HUMAN-CENTERED

Open Blueprint is a trademark or registered trademark of the IBM Corporation.

IBM encourages companies to use the Open Blueprint as a model for the development of their information technology architectures. The content and concepts within the Open Blueprint also can be applied to many client/server or distributed systems related information technology activities.

The following terms are trademarks or service marks of other companies:

AT&T	American Telephone & Telegraph Company
CompuServe	CompuServe
CORBA	Object Management Group
Digital	Digital Equipment Corporation
DOS	Microsoft Corporation
Encina	Transarc Corporation
Ethernet	XEROX Corporation
Hewlett-Packard	Hewlett-Packard Company
HP	Hewlett-Packard Company
IPX	Novell, Inc.
Kerberos	Massachusetts Institute of Technology
MCI	MCI Communications Corp.
Micro Focus	Micro Focus
Motif	Open Software Foundation, Inc.
NCP	Novell, Inc.
NetWare	Novell, Inc.
NetWare Core Protocol	Novell, Inc.
NFS	Sun Microsystems, Inc.
Novell	Novell, Inc.
ObjectStore	Object Design, Inc.
OpenDoc	Component Integration Laboratories
Open Software Foundation	Open Software Foundation, Inc.
OSF	Open Software Foundation, Inc.
OSF/Motif	Open Software Foundation, Inc.
Palladium	Massachusetts Institute of Technology
PCL	Hewlett-Packard Company
PostScript	Adobe Systems Incorporated
Taligent	Taligent
UNIX	X/Open Company Limited in the U.S. and other countries
XTI	X/Open Company Limited in the U.K. and other countries
X/Open	X/Open Company Limited in the U.K. and other countries
X/Open Transport Interface	X/Open Company Limited in the U.K. and other countries
X Windows	Massachusetts Institute of Technology

About This Book

This book provides a technical overview of the Open Blueprint. The Open Blueprint is a structure for 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*, G326-0395, which provides a higher-level description of the Open Blueprint.

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
- 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, 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

A bibliography is also included.

What's New in this Edition

This edition describes the newly enhanced version of the IBM Open Blueprint. The enhancements are as follows:

- Expanded description of the support and usage of object-oriented function within the Open Blueprint.
- Inclusion of Asynchronous Transfer Mode (ATM) as a subnetworking technology.
- Addition of Signalling and Control Plane to support usage of ATM and Telephony functions.
- Additional Application Services resource managers: Event Services, Compound Document, Collaboration, Telephony, and Digital Library.
- Additional Data Access resource managers: Hierarchical Database, Object-Oriented Database, Persistence Service, and Storage Management.
- Updated descriptions of each of the previously described resource managers.

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 74 for ordering information.

Chapter 1. Overview

IBM's Open Blueprint 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 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 the 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[2]; the security and directory protocols and usage as per the Open Software Foundation Distributed Computing Environment (OSF DCE); 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 (OMG CORBA) and the Common Object Services Specification (COS).

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 is the structure that enables the network of operating systems to function as a unit, as a **network operating system.** A network operating system is comprised of multiple systems that are separated from each other and are connected by a communication network. In the network operating system enabled by the Open Blueprint, 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 19.

Just as an operating system provides the management of resources on a single system, a network 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 and client/server applications. Figure 1 on page 2 represents one instance of a system in the network operating system. Figure 2 on page 4 represents a network 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.

2 Originally defined by the IBM Networking Blueprint.

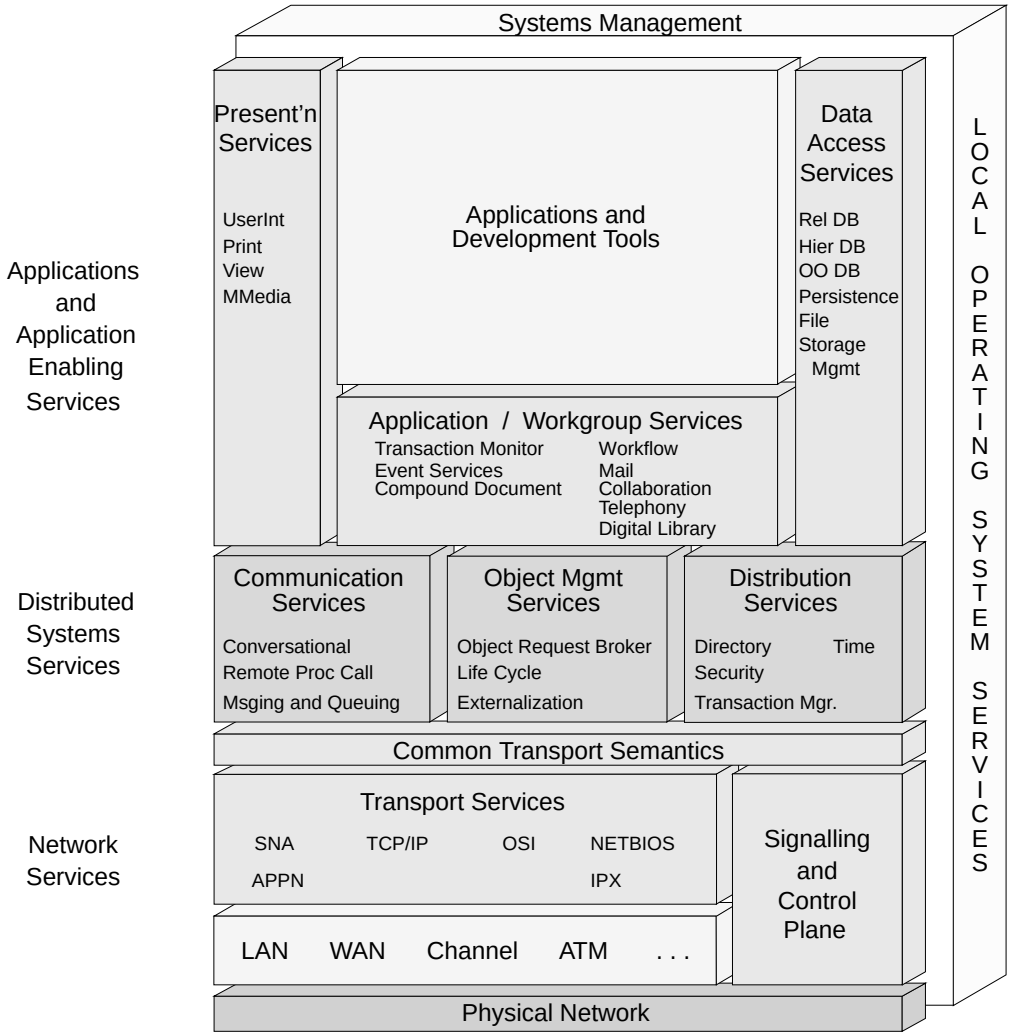

Figure 1. The Open Blueprint

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

Network Services

- **Common Transport Semantics** support protocol-independent communication in the distributed network.
- **Transport Services** provides the protocols for transporting information from one system to another, such as SNA/APPN, TCP/IP, OSI, 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 mechanism for interaction between applications and the user, using such graphical user interface elements as dialogs, icons, and menus.
- **Application/Workgroup Services** are common functions, such as mail, 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 for managing the network operating system by a system administrator or by automated procedures.

Local Operating System Services operate within a single system in a network. Examples of local services are managing memory 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 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. 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 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 Distributed Computing Services (XDCS) Framework and OMG's Common Object Services Specifications (COS).

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 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.

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.

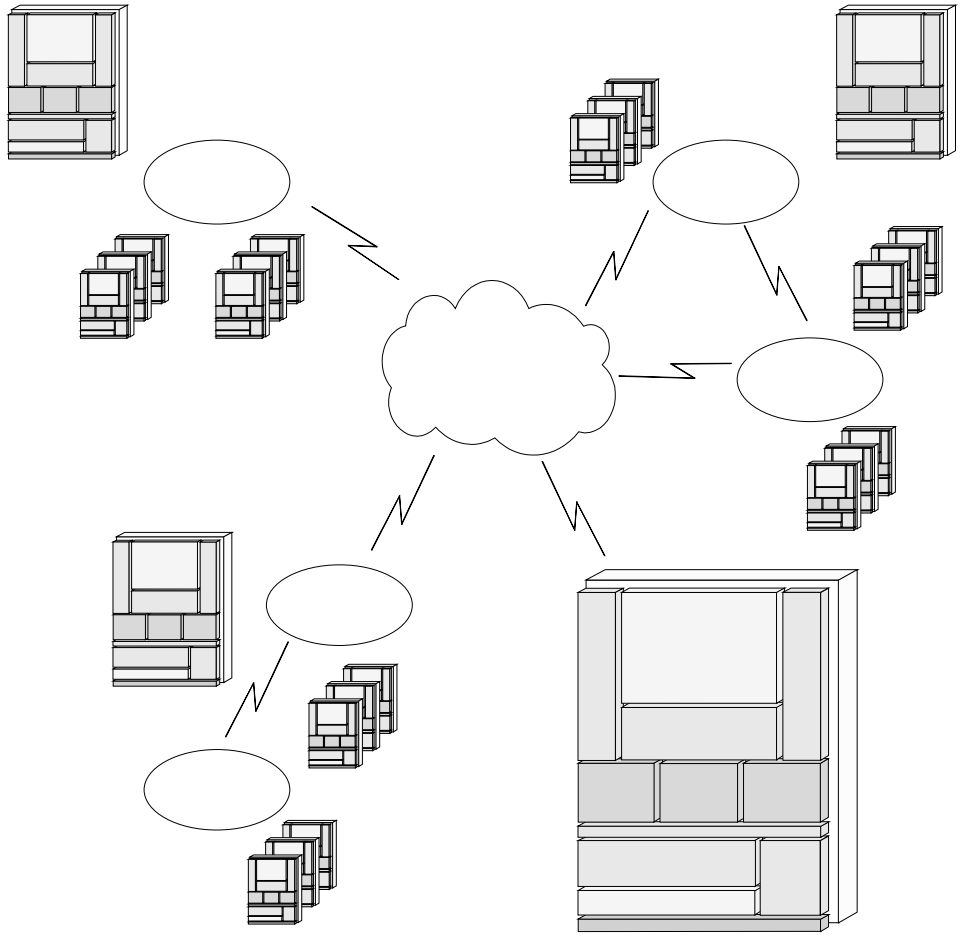

Figure 2. Network Operating System

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 instead 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, file system, and 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. However, 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 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, sharable 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.

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 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. 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.

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.

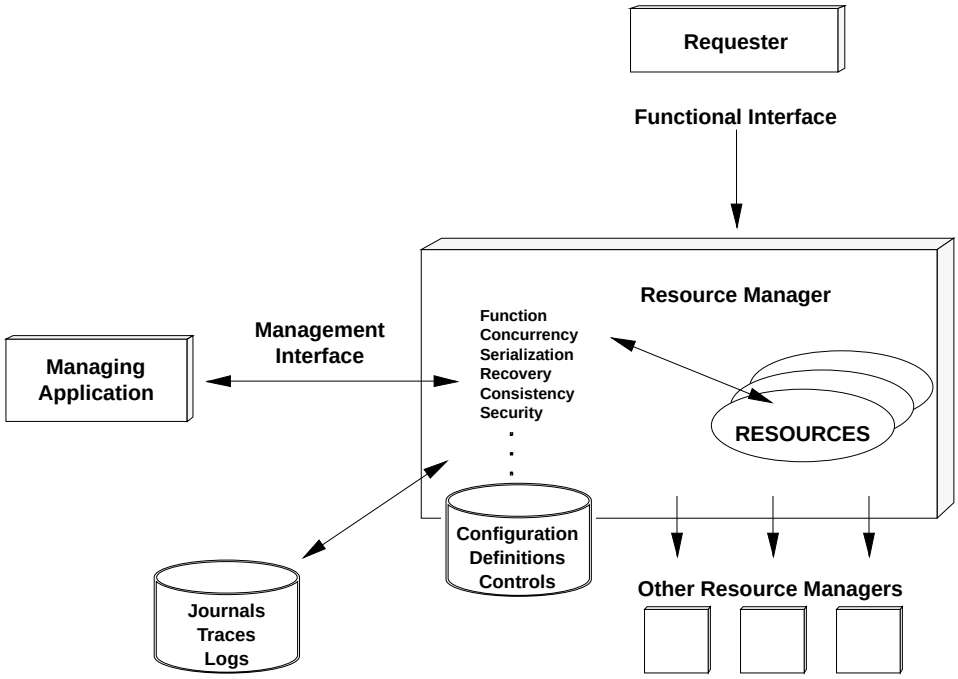

Figure 3. Resource Manager Characteristics

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; no operations 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, and by other resource managers[3].
- **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.

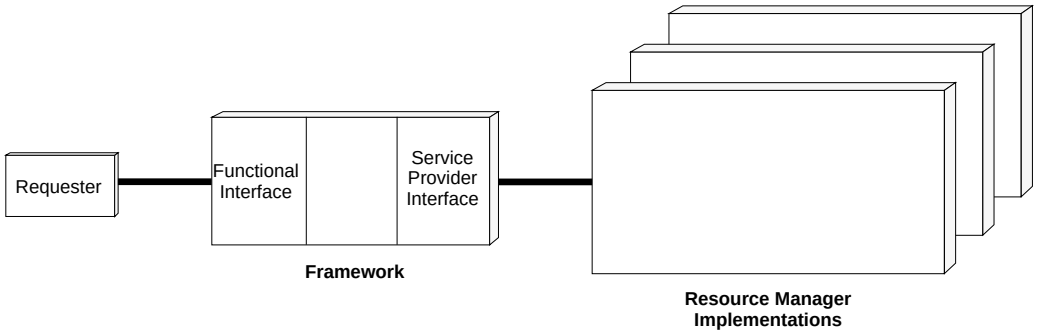

Figure 4. Resource Manager Interface Frameworks

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.

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 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.

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

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 61 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 resource manager.

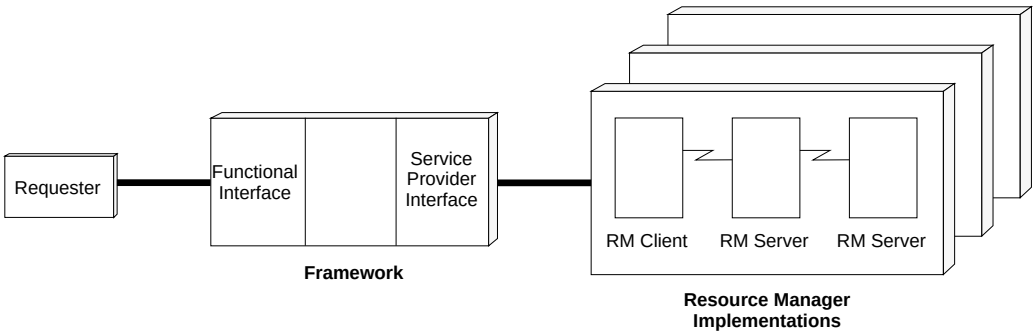

Figure 5. Open Blueprint Distributed Resource Manager Structure

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[4].

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.

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.

---

[4] **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**.

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 name space 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 the System Object Model (SOM) 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 for the System Object Model conforms to the Object Management Group's Common Object Request Broker Architecture (CORBA).

Object management services also include Externalization and Life-cycle Support Services in support of object-oriented programming.

As objects communicate with each other, certain Open Blueprint services may be applied implicitly as determined by the way the object classes were initially defined through SOM. These services are provided by the appropriate resource managers such as Persistence Services, Transaction Manager, and Security. Some systems management services, such as licensing and usage accounting may also be implicitly applied.

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**.[5]

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

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.

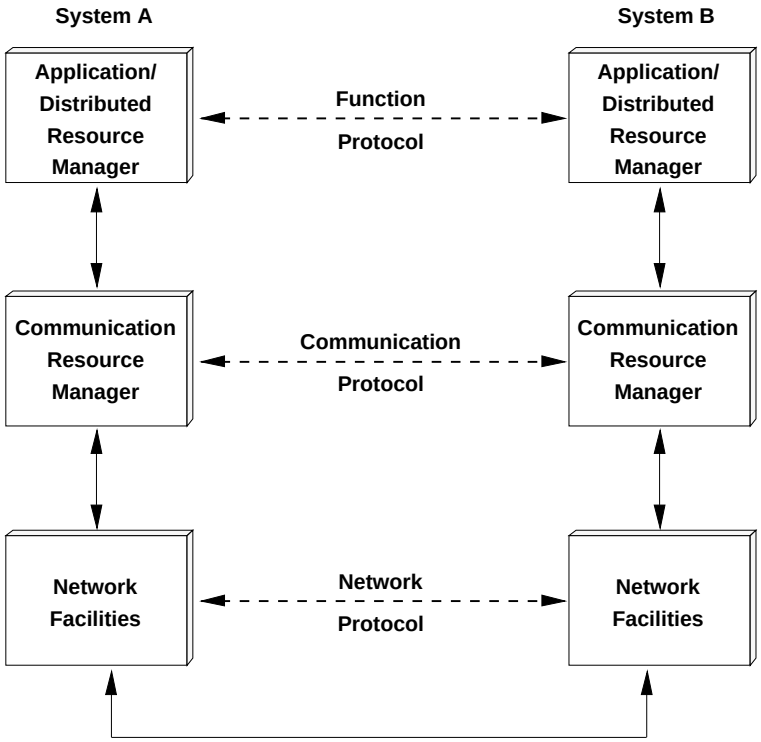

Figure 6. Protocol Layers

Normally, when two components 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 19). 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.

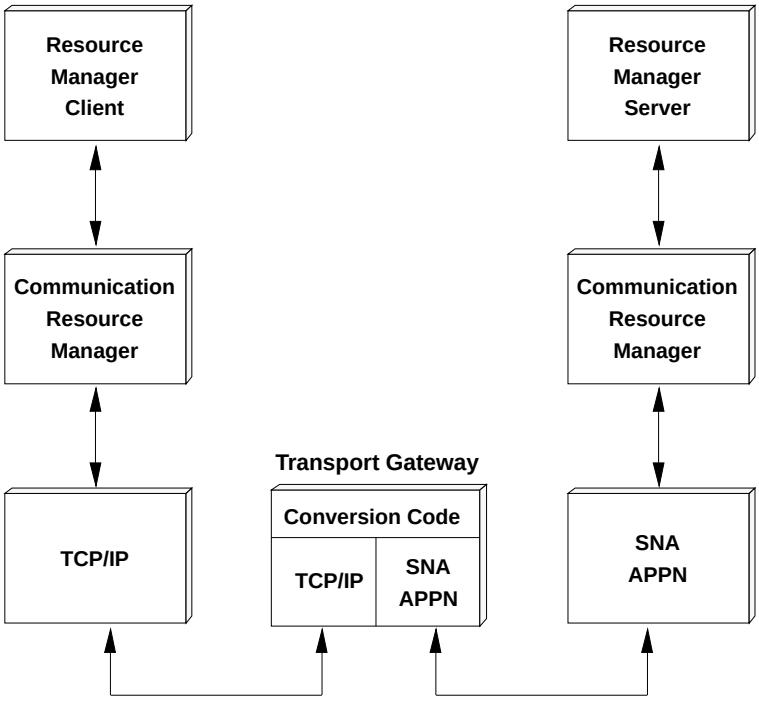

Figure 7. Role of a Transport Gateway

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 54.

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 Services resource manager, with at least one network driver. For every system it communicates with, it must share a common driver type, or share a common driver type with another system that contains a transport gateway to the desired end point.

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

The Database, File, Print, and Transaction resource managers have prerequisites of either the Conversation resource manager or RPC. Object-oriented resource manager implementations require the Object Management resource manager.

Client Platforms: 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. Smaller systems, like mobile computers or personal digital assistants (PDAs), could be configured to contain only a few resource manager frameworks and requisite 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. For an Open Blueprint distributed system to be functional, the server parts of certain critical resource managers (like Directory, Security, and Time) must be accessible somewhere in 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:

- 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 because they can provide the best function and interoperability demanded by customers. IBM products will provide and exploit standard interfaces and protocols as described in the Open Blueprint; 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 67 for a list of key documented standards for interfaces and protocols.

Integration

The Open Blueprint 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 promotes the integration of a broad range of client/server products under a single, network-wide signon process, a single name-resolution directory for locating distributed resources, and common services and graphical user interface for administration. Through the use of Open Software Foundation (OSF) Distributed Computing Environment (DCE) technology for the underlying directory and security services, the Open Blueprint also promotes interoperability with other vendor-provided products.

Structure for Heterogeneity

The Open Blueprint 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 8 emphasizes the support for multiple protocols.

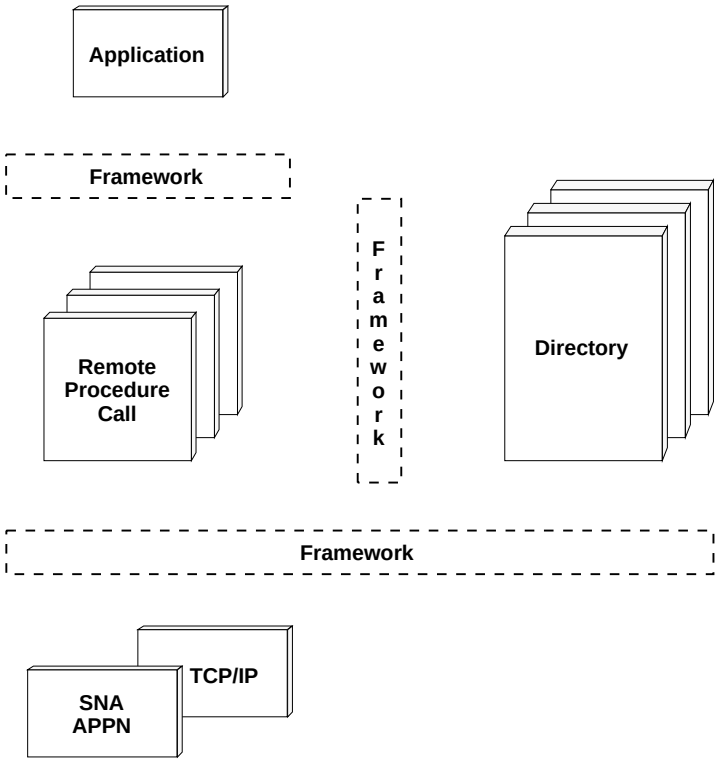

Figure 8. Structure to Support Multiple Protocols

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.

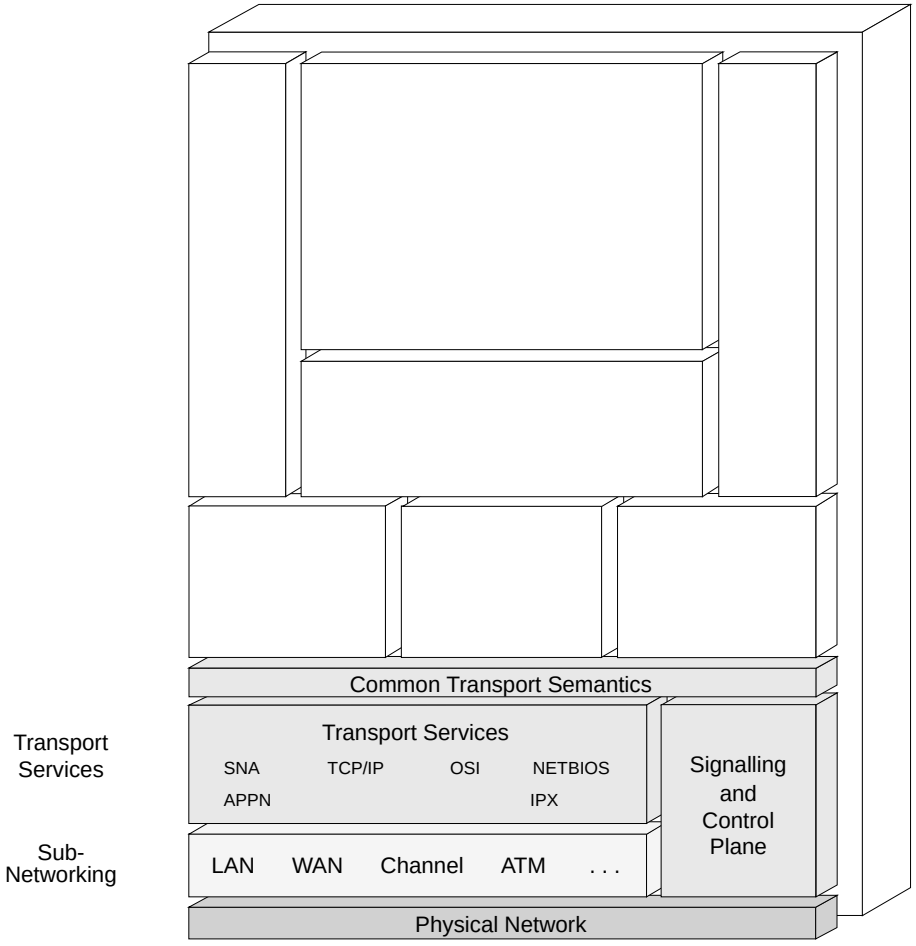

Figure 9. Network Services in the Open Blueprint

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 9, 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) without any change.

Using Common Transport Semantics enables the integration of networks with different protocols through transport gateways. As shown in Figure 10 on page 25, 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.

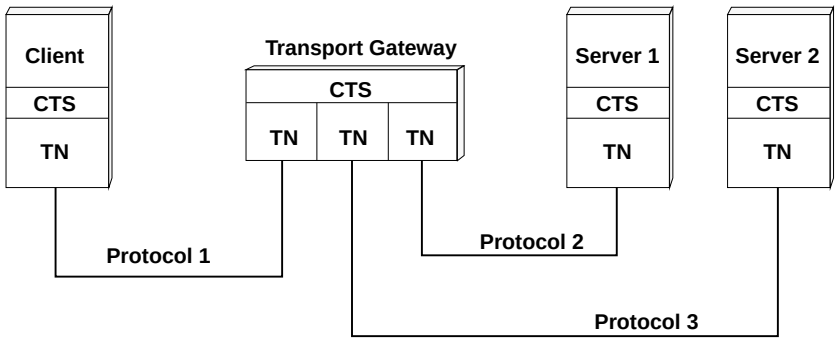

Figure 10. Multiprotocol Transport Gateway

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

The Transport Layer Protocol Boundary (TLPB) defined by the Multiprotocol Transport Networking (MPTN) architecture and implemented in AnyNet products, is at the top edge of the Common Transport Semantics. Sockets, CPIC, and RPC interfaces can access the multiprotocol transport network through the TLPB.

The TLPB protocol boundary enables higher-level Communication Services (such as RPC or Conversational) to support multiple transport protocols transparently[6]. It also gives applications and higher-level resource managers the choice of achieving transport independence by using the TLPB directly, where appropriate, 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 does not shield applications or services from network differences. IBM has developed AnyNet products (based on MPTN architecture) that shield requesters from network differences. X/Open has published preliminary 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[7]
: 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 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

6 This means that the using system may continue to use its native form of network addressing (such as SNA, NetBIOS, or INET) though the TLPB.

7 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**.

could be applied within the transport system. It also permits other delivery semantics, such as multicast (sending a packet to many receivers).

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:

- Systems Network Architecture/Advanced Peer-to-Peer Networking (SNA/APPN)
- Transmission Control Protocol/Internet Protocol (TCP/IP)
- Open Systems Interconnection (OSI)
- 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.

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 scaleable 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 three common distribution models. Each model describes how distributed parts of applications or resource managers communicate with one another. They are:

- Conversational
- Remote Procedure Call
- Messaging and Queuing

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 SNA APPC, where one part of a distributed application or resource manager initiates a conversation[8] with another, and they then exchange messages, synchronizing as necessary, until the user's requests are satisfied, at which time the conversation is terminated. 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, which is based on the SNA APPC architecture. X/Open's Common Programming Interface for Communications (CPI-C) provides a common interface for the implementation of the conversational model on all major platforms for access to both LU6.2/APPC conversations and OSI-TP dialogs.

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.

---

[8] If a session had not been established previously, this must occur first. This involves establishing a logical connection with the distributed application or resource manager at the target server.

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 it 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.

Selecting a Communication Resource Manager: Although it is likely that almost any distributed function **could** be implemented using any of these 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 communicate, where the calling program is not blocked once 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.

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

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 **operation call** in such a manner that a number of technical characteristics of object technology, such as encapsulation, inheritance, and polymorphism, are supported.

The Open Blueprint Object Request Broker incorporates IBM's System Object Model (SOM) and related distributed SOM technologies[9] that address the inherent interoperability problems of objects. Since each object-oriented language has a unique set of object characteristics or a unique object model, it has not been possible to write parts of an application in one language (C++) and use these parts from another language (Smalltalk). Different compiler implementers for the same language (C++) have implemented different foundation mechanisms. The objects contained in a binary module compiled by one compiler cannot be used by a using program compiled by another compiler.

SOM provides a rich, language-neutral object model that supports binary interfaces defined at the system level and independent of a particular compiler implementation. The SOM object model maps easily into the popular object-oriented programming languages and the 3GL language base used in existing systems.

Distributed object support associated with SOM provides a complete implementation of the industry standard Object Management Group's Common Object Request Broker Architecture (CORBA). Because of the seamless integration, CORBA's standards apply equally to both local and distributed objects. The Open Blueprint Object Request Broker provides CORBA-compliant support for distribution of objects, and through the use of the Messaging and Queuing resource manager, SOM allows asychronous communication between the using program and the target object such that both do not have to be active at the same time.

---

[9] The initial delivered support for this function is named DSOM.

A key advantage of SOM and distributed SOM is location transparency. The same mechanism that provides access to local objects is extended to access remote objects. As illustrated in Figure 1 on page 2, the Object Request Broker operates above the Common Transport Semantics in the distributed system so that this technology is available to configurations involving a variety of transports. The distribution services are used so that this object service operates as an integrated part of the overall network. This includes using the distributed directory to access information about target objects, the security services to authenticate users, and the Transaction Manager to synchronize object operations with other resource requests.

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 services and Externalization 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.

In addition to these services, OMG is continuing to define **Identity** and **Collections** services. The Identity services makes it possible to determine whether two objects are really the same object. The Collections services provides basic capabilities of 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.

Today's operating systems frequently define one or more name spaces unique to the system. These name spaces may 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: A federated[10] namespace is the logical union (via a technique called junctions) 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 structure implemented in OSF's DCE technology. 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.

[10] Federated naming combines the concepts of both universal and concatenated naming as described in the previous edition.

The federated name space includes two classes: global and cell. Resource manager namespaces may be federated in the cell namespace. This allows a resource manager's resources to appear as part of the federated namespace via a federated name. Names in the global namespace use the ISO X.500 or the Internet[11] naming standard. Names in the cell namespace follow the Cell Directory Service (CDS) naming conventions (untyped) as defined by DCE. Names in a resource manager namespace follow that resource manager's naming conventions. Figure 11 shows examples of federated names.

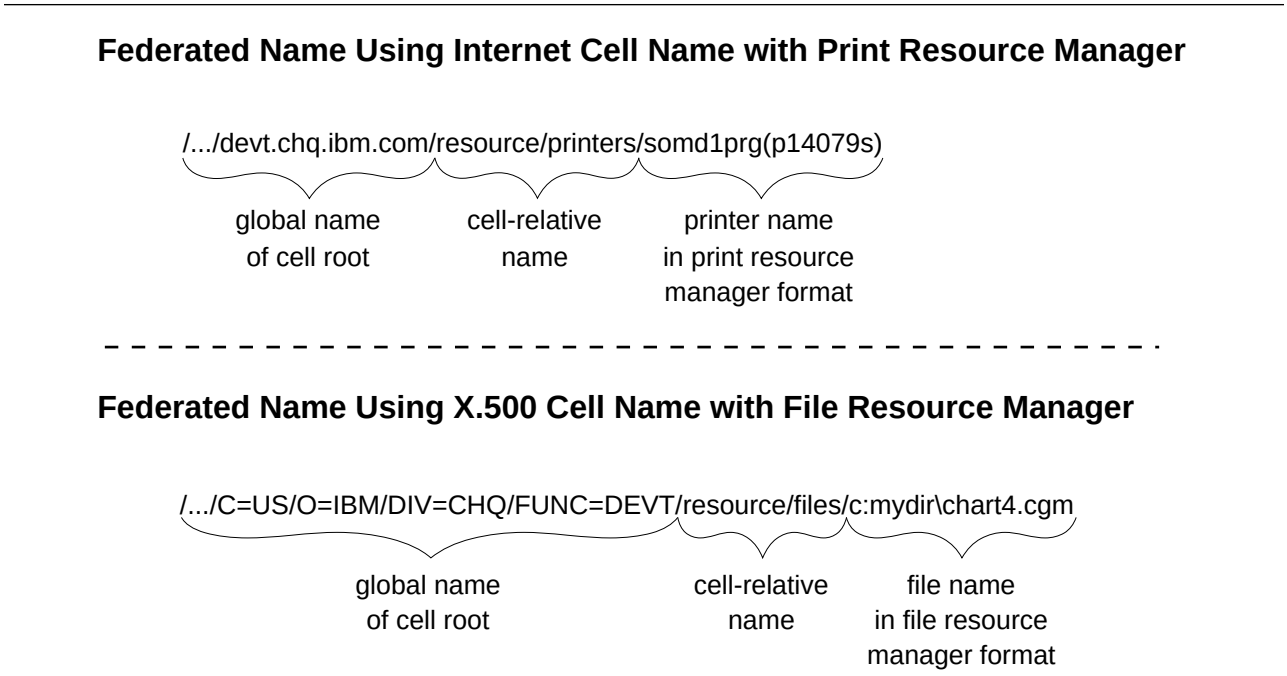

Figure 11. Federated Naming Examples

Over time, new resource managers, APIs, user interfaces, tools, and application programs will support the federated name space. Existing system and resource manager-defined name spaces 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 name space 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 name. 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 name space. 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.

[11] Internet names are handled by the Domain Name Service (DNS), which is a standard TCP/IP application.

Directory: The Open Blueprint Directory provides a database of information about resources in the distributed system. Since resources in the distributed system follow standard naming conventions, passing these names to the directory services allow 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. Hence, client functions can dynamically learn the location of and how to access a resource, without it having been previously defined as part of the configuration.

The Open Blueprint Directory is based on X/Open Federated Naming concepts and Open Software Foundation DCE directory technology. The Open Blueprint Directory Service 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 Service 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 Service. Given an attribute or set of attributes, a name or set of names satisfying those attributes may be returned. For example, given the name of a print server, the Directory Service can return the printer's location. Or given a print location, the names of print servers for that location may be returned. The Directory Service 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 servers: the global directory service (GDS), which may be either an X.500 or an Internet Domain Name System (DNS) directory service, and a cell directory service (CDS)[12]. X.500 and CDS support distributed naming and server replication for backup or performance optimization. Resource manager specific directory services and associated namespace may be federated in the cell directory namespace.

CDS uses RPC for all client-to-server protocols (all name resolution services) and server-to-server protocols (such as replication). X.500 uses communication protocols defined by OSI standards for both client-to-server and server-to-server protocols. DNS uses communications protocols defined by the IETF for client-to-server protocols. In the Open Blueprint, both CDS and GDS protocols flow across multiple underlying transports using Common Transport Semantics.

There is a naming framework called the Extended Naming Context (ENC) derived from the OMG Naming Service that presents the application with a set of strategic application programming interfaces (APIs) and allows a directory service to be plugged into this framework. The framework implements federation and allows the application to be written independent of the directory service. The API set consists of an object-oriented interface based on the OMG Naming Service, X/Open's Directory Service and Object Manager (XDS/XOM), and the Name Service Interface (NSI) which is part of X/Open's Remote Procedure Call interface.

Directory User Interface: The Open Blueprint includes a user interface for storing, retrieving, and searching information in the name space. This graphical user interface provides capabilities for managing information stored in GDS, CDS, or other federated namespaces such as the security registry. In addition, it is an open structure that can be extended to allow resource manager-specific information to be

[12] Based on DCE directory technology.

interpreted and presented. Open Blueprint resource managers provide the extensions necessary to enable their unique directory information to be viewed and manipulated by users.

Security: Security is concerned[13] with identification and authentication of users, access control to resources, integrity of information, confidentiality of information, audit facilities, and management of security information. Security services are provided by several distributed resource managers.

The Identification and Authentication resource manager verifies the identity of a user or server. The Access Control resource manager provides access authorization services. The Audit resource manager allows the specification of audit activities and collection of audit data. Other services such as Information Integrity, Information Confidentiality (or privacy), and Non-Repudiation are made available through the Identification and Authentication resource manager API.

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 is unforgeable and unique across both space and time. Such identifiers are called Universal Unique Identifiers (UUIDs).[14] These unforgeable identities are not user-friendly and, therefore, principals will also have user-friendly names such as JOE or WILSON associated with them. In the distributed system, security decisions are made using UUIDs because user-friendly names may not be unique. The binding between UUIDs and the user-friendly names is maintained in the security database, or security registry.

Note: This book uses the term “user” instead of “principal,” except when the context requires “principal.”

Identification and Authentication: Users can potentially access the distributed system through many client systems. Authentication can be done through any of those systems. Authentication must be done before the use of any system service.

Authentication of a user proceeds in two steps. Local authentication involves validation of the user identity by a local identification and authentication resource manager (where one exists) using traditional approaches. For network authentication, identity validation with the network Identification and Authentication resource manager is also required.

- Network authentication begins by first locating (using the distributed directory) the appropriate Identification and Authentication resource manager instance that can validate the identity of this user.
- Sufficient identity-based information is then sent to the Identification and Authentication resource manager so that it can verify the user's identity and determine the user's **credentials**. Credentials are the set of information in addition to the user's identity that can be used by the Access Control resource manager to make access decisions. It is expressed as membership of specified groups, such as Administrator, IBM Employee, or Dept 605[15].

---

[13] The basis of computer security is some level of physical security and software system integrity within any particular computer. The Open Blueprint does not address the provision of this security but assumes that it can be provided. The Open Blueprint addresses the distributed system issues including the interoperation of systems having high levels of security with those having none.

[14] The UUID is a computer-oriented identification scheme that provides unique identifiers. UUIDs can be generated efficiently in many different places in the distributed system without ever having the same value generated more than once. This assumes that the UUID generator is not tampered with, works correctly, and that the “seed” provided for the generation is properly defined. The UUID is a 16-byte value that is opaque to all users. The only valid operation on UUIDs is a test for equality. The 16-byte UUID permits every person in the world to have a computer that generates a UUID every microsecond for three million years assuming an allocation mechanism that does not waste values.

[15] The credentials actually contain the UUID of each group, not a string name.

Once local and network authentication are performed satisfactorily, the user will not have to be authenticated again for the duration of the session no matter how many local or remote resource manager services he or she chooses to access.

Successful completion of local and network authentication is usually referred to as the equivalent of the successful completion of **logon** in standalone systems. Because the distributed system will not require any further identification from the user, it also possesses the property of **single signon**, a key user requirement[16].

Open Blueprint network authentication is based on the OSF/DCE technology which in turn was derived from the Massachusetts Institute of Technology's Kerberos technology. The functions of the Identification and Authentication resource manager are accessed through the GSSAPI. In cases where logon to remote non DCE-aware applications is required, DCE technology is complemented by an IBM-developed technology for third party authentication[17] that extends the single signon paradigm to support remote direct usage. Both technologies are supported by security attributes stored in the DCE Registry database, and the provision of credentials by the DCE Identification and Authentication resource manager.

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 services, 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[18].

Secure transmission of the user credentials between nodes is done through one of the Communication resource managers. The integrity of these credentials is also protected by the Communication resource managers during transmission. Integrity protection enables the recipient resource manager to detect any possible malicious or unintentional modification of the credentials during transmission. This is done by embedding credentials in a data structure called a **certificate**. Certificates cannot be fabricated by any untrusted party[19].

The Identification and Authentication resource manager can (upon user request) construct certificates that permit a user to pass on his privileges to a receiving server so that it can gain access to another server. For example, a user who wants to print a file may pass his or her right to read that file to the print server, so that the print server can read the file in order to print it. When a user permits another principal to function on its behalf, possibly with some restrictions, to access a specific service, it is called **delegation**. When a user assumes the credentials of another user entirely with no changes, it is called **impersonation** or forwarding.

Information regarding logon and logoff activity of users will be stored by the Identification and Authentication resource manager and may be distributed to appropriate resource managers. This knowledge is essential for products such as office systems that must be able to deliver mail to an individual, and the Workflow resource manager that may need location information to distribute pieces of work. Location information cannot be relied upon because there is no guarantee that the user is still at the location by the time the information is used. Resource managers using location information must therefore use the authentication mechanisms to ensure that the user with whom communication is established is the one expected.

---

16 Server/machine authentication follows the same pattern except that server passwords are stored suitably protected on the system—in a file for example—so that only the server logon code has access to the data.

17 KryptoKnight, as implemented in NetSP.

18 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.

19 If there are no untrusted parties in a position to fabricate the certificate, the certificate can be the user's identity. When the path is not completely trusted, cryptographic certificates are required.

Information integrity and **confidentiality** (when information is in storage and during transit) is achieved by invoking encryption services. 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 "signature." Client code or users are able to specify the options they want to use as well as provide appropriate keys for decryption. Typically, these services are provided transparently by Communication resource managers.

Access Control: The Access Control resource manager determines if a user is allowed to do what he or she is asking to do. 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.

A syntactically and semantically common form of Access Control List is defined so that resources can be moved between systems, along with their access control information, without changing the security control over them. The common form of the Access Control List is the **OSF/DCE ACL** with the direction being to migrate to the POSIX ACL when it is defined[20]. Resource managers use the Access Control resource manager where it is appropriate.

Audit: The Audit resource manager provides an audit API which is based on extensions to the POSIX audit APIs. The Audit API 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. Audit services are administered through a special Graphical User Interface (GUI) from an administrator console as part of security administration facilities. A set of auditable events is provided from which the administrator can select appropriate activities to be audited.

The Audit resource manager supports access 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, and/or with a number of local security resource managers. A key attribute of security administration is that it can be performed within administrative scopes that are chosen by the customer. 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.

[20] This is the OSF direction also.

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[21]. 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[22]). The Time service 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 dicontinuities 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.

---

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

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

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.

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

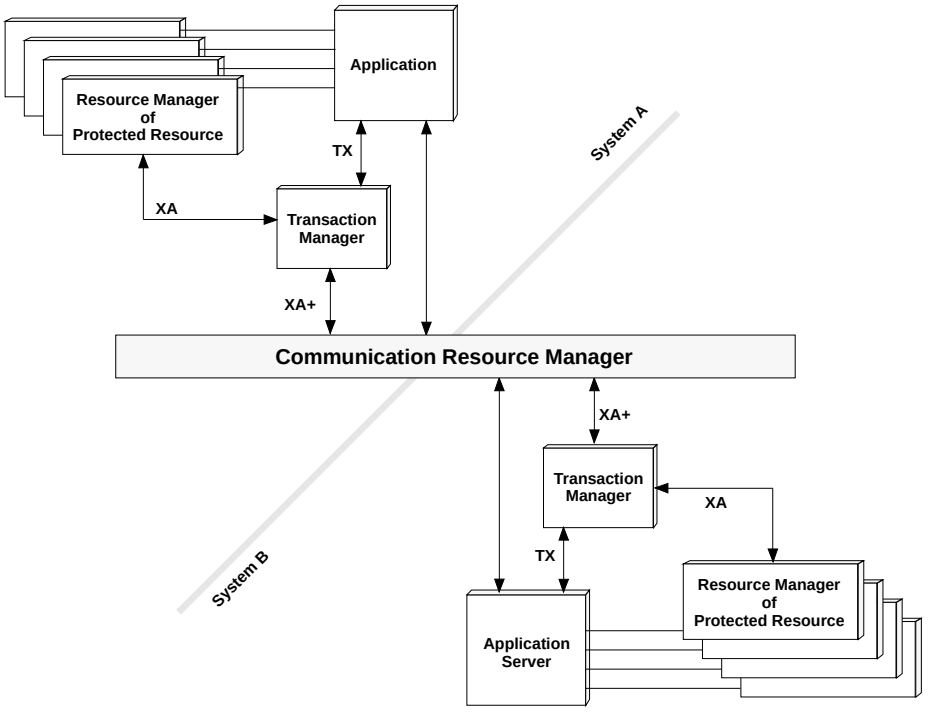

Figure 12. Transaction Manager/Resource Manager Relationships

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 12, 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.

The Transaction Manager also supports the OMG Object Transaction Service interfaces.

The X/Open standard definition of the procedural interfaces is the basis for future technical work in transaction management. Current transaction management products will continue to support their existing interfaces which are functionally-equivalent to the X/Open interfaces. The functions of these current interfaces have served as a technology base for the X/Open transaction manager support.

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.

Directory User Interface
Used by resource managers to store resource manager-specific information in the directory and enable it to be viewed and manipulated through the graphical user interface (see “Directory” on page 33). Alternatively, the resource managers may use directory services interfaces to store resource manager-specific information in the directory.

Security Services used to obtain the credentials of the originating user[23], 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

[23] 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.

turn interfaces with the Transaction Manager in the remote system to pass it the synchronization request[24].

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

[24] 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.

Applications and Application Enabling Services

Presentation Services

User Interface: The User Interface 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 of the User Interface resource manager 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 **user interface** describes the front-of-screen appearance and function of an application or system. 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 space is a desktop and his/her 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 Macintosh's Finder, Windows' Program Manager, OS/2's Workplace Shell, and AIX's CDE. Emerging natural computing technologies such as speech, multimedia, and agents challenge the traditional desktop to evolve into something even more powerful and intuitive. Key to the success of this new generation of desktops is a new, compelling user interface metaphor, supporting the notion of customizable, specialized user environments, where the focus is always on maximizing the value of the end user experience, over and over again.

Ideally, a desktop serves as a base environment into which applications can be “plugged.” The desktop “substrate” provides for drag-and-drop, embedding and linking, dynamic data exchange, installation and configuration, help, and many other functions shared by applications. These facilities provide the ability to integrate application work on the desktop. For example, a change in one application results in a function occuring in another application, or a change to data in one application is reflected in the data seen in another.

The User Interface resource manager uses the following technologies:

- OpenDoc[25] compound document technology is the basis for the development of all parts and places, and is used to manage the user's screen.
- OpenDoc linking and embedding technology is the vehicle that enables parts to be interchangeable, shareable, and portable.
- OpenDoc Open Scripting Architecture (OSA) scripting frameworks provide user and programmer control over parts and places.
- SOM is the under-lying, low-level object model providing the capabilities of binary release-to-release compatibility, multiple language support, and OMG CORBA compliance.
- Communication between objects in separate processes is achieved through distributed SOM and related distribution services.
- Platform-specific windowing and graphics systems provide underlying support for the user interface elements.

[25] 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.

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

Task-oriented user environments are based on **places** which replace the current, single desktop metaphor. A place contains people, things, and other places necessary to accomplish a specific task or set of tasks. Places support sharing and collaboration between users. Multiple users can 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. A place is distinguished by its content, visual appearance, and user interaction paradigms. Places are compound document (OpenDoc) parts and contain things that are OpenDoc parts. Places and the things in them embrace the latest in natural computing technologies to enrich the user interface.

There are several levels of parts used in the desktop environment. The Compound Document resource manager provides a set of core parts from which other documents/parts can be composed or specialized. The User Interface resource manager provides a set of desktop parts from which the content of a place begins to materialize. For example, a home computing place may have parts for an address book, a check book, an entertainment guide, and a telephone. All these parts are in turn combined into "heavy-weight" parts which are full-function, compound document-based applications aligned with the tasks users wish to perform.

Desktop environments in support of user needs are evolving to accommodate a proliferation of new user interface metaphors. The days of the single, "messy desk" metaphor are coming to a close. Instead, desktop environments are becoming more modular and configurable. The User Interface resource manager uses key component technologies to establish the base on which specialized desktops and places can be built.

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, and 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, 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 (queues, jobs, printers, etc.), 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, enabling 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[26] and job information (for example, scheduling and media specifications) to be printed.

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 implemented using RPC[27]. Line printer controller and spooler/line printer daemon (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 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; e.g., 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 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

---

[26] 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.

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

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)**.

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 and facilitates workgroup collaboration.

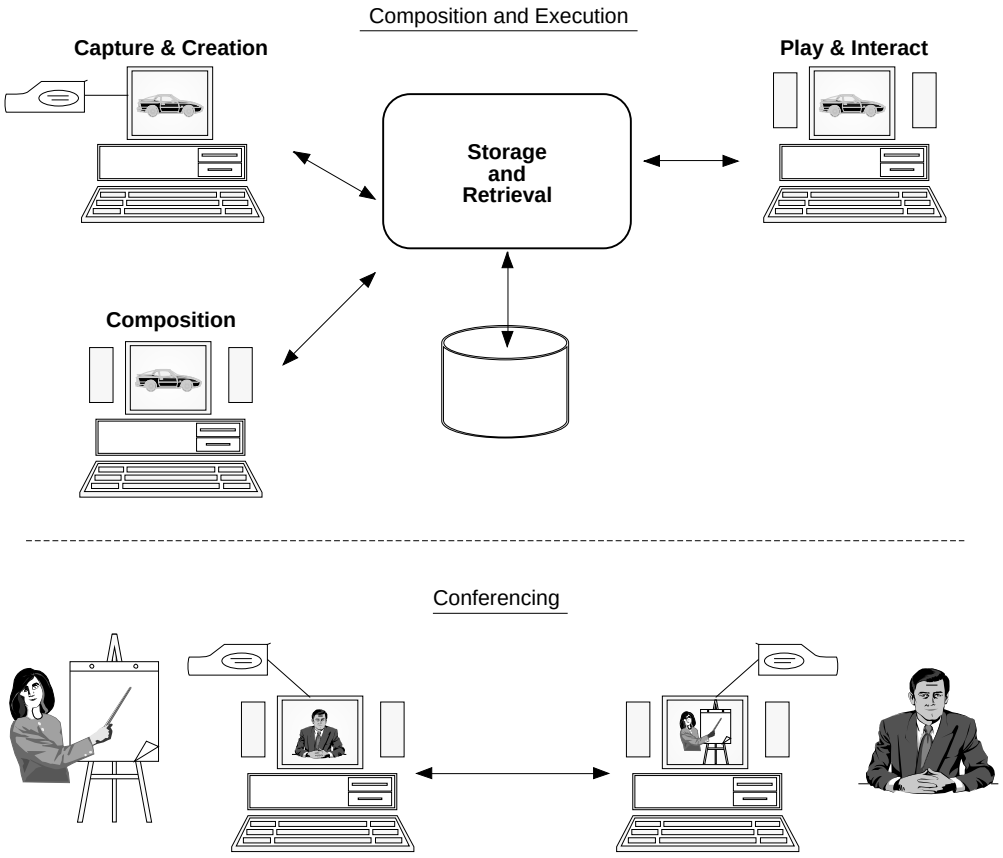

Figure 13. Generic Multimedia Application Capabilities

Figure 13 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. A key multimedia application is audio/video conferencing in support of workgroup collaboration.

In the Open Blueprint, multimedia services provide an environment in which a heterogeneous set of multimedia-enabled system platforms cooperate to service interactive multimedia application requests to control and synchronize data steams that flow from source to destination devices that can be located

anywhere in the network. Multimedia resources are classified into multimedia logical device types: multimedia file, video camera, microphone, video window, and so forth. A multimedia logical device is a generic representation of a class of multimedia devices available from different vendors. In the simplest distributed multimedia computing environment, a multimedia data stream flows from a source multimedia logical device over a logical connection to a destination multimedia logical device. Generally, the source and destination multimedia logical devices are associated with different distributed system nodes. Sometimes a complex configuration of interconnected multimedia logical devices is required, as in the case where audio and video data streams need to be compressed and combined (multiplexed) at the source to preserve synchronization and sent over a network connection and then split (de-multiplexed) and decompressed at the destination. Audio/video conferencing is a good example of an application requiring a complex configuration of interconnected, multimedia logical devices representing each participant's camera, microphone, headset, and display.

Multimedia data may be played from a multimedia file server using redirected file I/O or by using the Multimedia resource manager to control data streaming across a network. The latter is more general because it encompasses both multimedia data playback to a single user and to multiple users participating in an audio/video conference. The distributed multimedia services of the Multimedia resource manager are used by an application to control:

- a configuration of interconnected multimedia logical devices,
- the data streams that flow between interconnected logical devices,
- and the synchronization of data streams.

The Multimedia resource manager employs the services of the Collaboration resource manager and through it manages those of the local operating system's Multimedia System Services.

Figure 14 on page 46 below shows how the Multimedia and Collaboration resource managers cooperate to provide distributed multimedia services to the generic multimedia applications shown in Figure 13 on page 44. The Collaboration resource manager is used to activate and deactivate multimedia logical devices (including the use of Multimedia System Services as a set of logical devices). The logical devices are interconnected via logical connections which provide a guaranteed quality of service. In addition, the Collaboration resource manager can set up logical connections needed to support the multimedia, multi-party communications requirements of audio/video conferencing applications. The operating system's Multimedia System Services manage the local multimedia resources which are the actual multimedia sources and destinations: a disk drive, VCR, audio cassette recorder, and so forth. The division of responsibility among the Multimedia resource manager for providing a common set of distributed multimedia services to applications, the Collaboration resource manager for managing logical connections that provide guaranteed QoS and interconnect multimedia resources, and the operating system's Multimedia System Services for managing local multimedia resources permits a broad range of multimedia applications to be supported in a distributed systems environment.

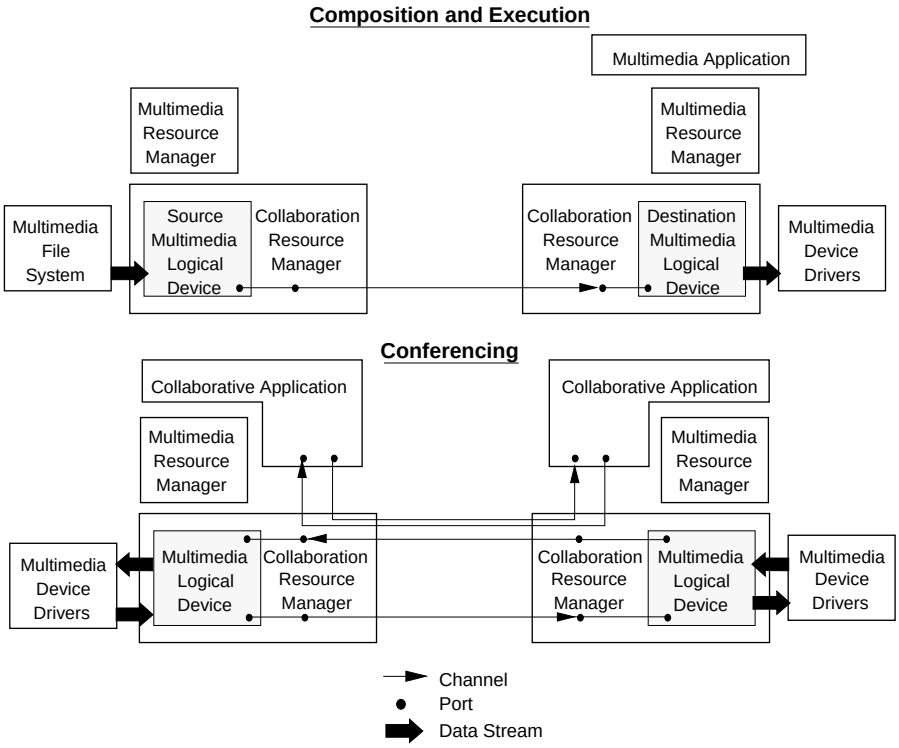

Figure 14. Distributed Multimedia Services For Generic Multimedia Applications

Application 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**). Additionally, 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 37) 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 applications themselves. This may be 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. Since 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. 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.

Event Services: The Event Services resource manager allows a program to send information (representing the ocurrence 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 conforms to 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 User Interface resource manager.

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

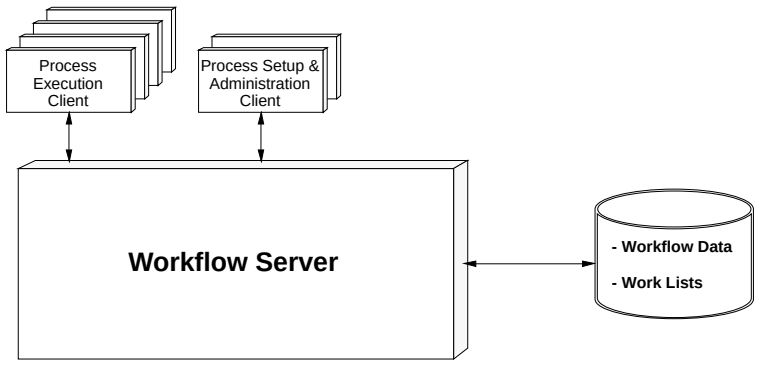

Figure 15. Workflow Manager Structure

The Workflow resource manager exploits the services of the other resource managers:

- Database and File, for access to databases and files.
- Digital Library, for access to forms, images and multimedia objects.
- Directory, to locate people, applications and configurations
- Security, for identification and authentication, authorization, 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 viewed not only as a mechanism for users to exchange text messages but also 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 electronic mail service 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 is widely used in today's networks. The Internet TCP/IP protocol supports a SENDMAIL service and Simple Mail Transfer Protocol (SMTP). Open Systems Interconnect (OSI) systems support the X.400 Messaging Handling Services protocol. Additionally, value added network providers like AT&T, MCI, and CompuServe support their own protocols and supply gateways to support X.400 or SMTP. IBM 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. In the Open Blueprint, Electronic Mail Servers will use the Messaging and Queuing resource manager for server-to-server message transfer and will natively support standard X.400, SMTP, and SNADS protocols. Access to other important proprietary message transfer standards will be through gateways.

Collaboration: The Collaboration resource manager serves two roles in the Open Blueprint.

It supports collaborative applications that allow people physically distributed over a computer network to work together in real time. Examples of collaborative applications are desktop conferencing and distance learning. Typically, the applications themselves are also distributed over the network and can permit for example, remote presentations and shared document editing among groups of people. Key aspects of support for collaborative applications include:

- **Programmable call control**: the establishment of a collective environment, akin to the call concept familiar in telephony, which sets the rules, or policies, for the interactions between the applications
- **Dynamic group participation**: the support of dynamically changing configurations and networks, reflecting the arrivals and departures of participants and possible communication links
- **Standards support**: In some cases, standards have been (or are being) developed to ensure interoperability over particular network types; such standards include the ITU-H.320 recommendations for data collaboration. These are defined with an emphasis on public switched data communication networks, such as ISDN. The Collaboration resource manager is designed to allow applications to be developed which are fully standards-compliant and therefore able to inter-operate with other compliant implementations.

A second role of the Collaboration resource manager is to support distributed multimedia through a connection model approach. This can be in support of audio-video conferencing or distributed multimedia playback. In response to application or Multimedia resource manager requests, logical communication channels (or pipes) are established. These channels are connected to logical devices which in turn support real devices such as cameras and speakers, as well as file systems containing multimedia objects. When logical devices are so connected, the data flows (streams) directly between the devices without requiring further application involvement.

Support for **Quality of Service** is a key function of the Collaboration resource manager. Quality of Service is the characteristic of a system that determines how successfully it can carry different kinds of information, and therefore its suitability for particular applications. As described on page 43, Quality of Service is particularly relevant to the support of multimedia data.

Quality of Service is an “end-to-end” requirement, typically from a local device to one or more remote devices distributed over a communications network. The Quality of Service function of the Collaboration resource manager is aware of the required resource commitments that must be made within the network nodes to satisfy the requested level of service. It establishes the connections only when sufficient resources, for example, processor cycles, bandwidth, and buffers are available to support the requested functions.

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.

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

Call Control encompasses basic first party and third party 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 third party 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 portion of the resource manager is connected with a telephone switch (for example, a PBX) through a computer-Telephony (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 first party 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. First party support will be consistent with industry initiatives.

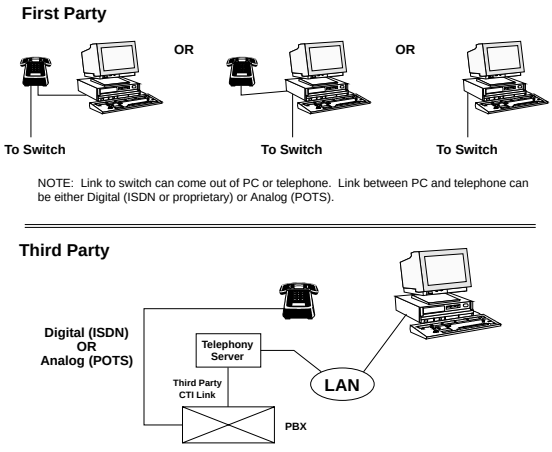

Figure 16. First and Third Party Telephony

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, place outgoing calls, etc. Through the use of voice related technologies such as speech recognition and synthesis, it can transform voice to text and text to voice.

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[28] 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 that is built on the base provided by the Open Blueprint security functions.

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

Data Access Services

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[29] 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:

---

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

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

- DDM[30] (Distributed Data Management) architecture for record file access.
- Distributed File System (DFS[31]) 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 17. The structure includes the possibility that a different file protocol is used between a client workstation and a file server, from that used between the file server and another file server. The first file server can thus shield the client from the multiplicity of file protocols, as well as possibly providing local caching of remote files, or exploiting a larger file server for archiving or back up of files.

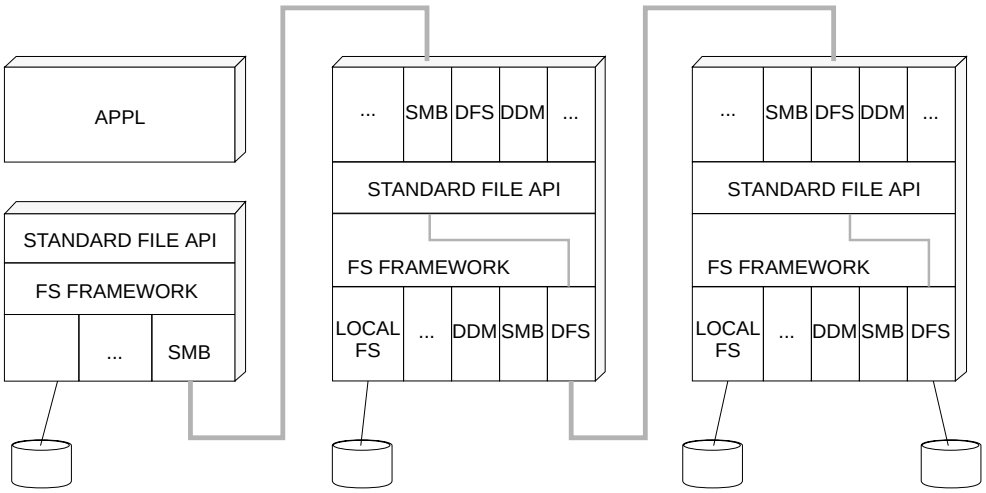

Figure 17. Support for the Heterogeneous File Environment

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.

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 ODBC interface defined by Microsoft for Windows. The CLI was developed because historically, application developers were forced to implement to a specific database client. The CLI permits a single application to work with any installed client,

[30] 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.

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

The RDB resource manager is a distributed resource manager. It supports clients[32] on machines with no local database. RDB resource manager servers have a local database and also cooperate with other RDB resource manager servers to satisfy a client's request. 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. The user or group identity to whom authorization is GRANTed is derived from a federated-named principal or group. Finally, instances of the RDB resource managers cooperate with the Transaction Manager on their local system 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.

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 18 shows the use of a functional protocol gateway to support heterogeneous, distributed data over any installed, supported transport.

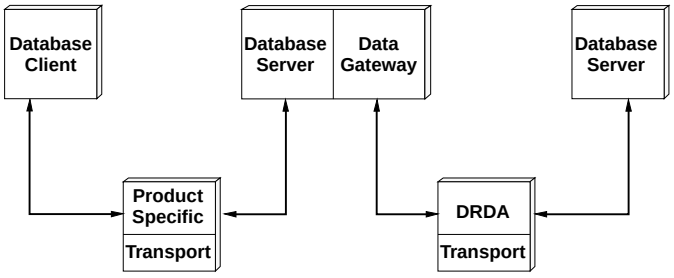

Figure 18. Non-DRDA Access to DRDA Data

The distributed RDB resource manager also provides support for data replication management for database information to allow the performance optimizations needed for an application to achieve near local performance in the distributed system.

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

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 and Network Support. 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++, SOM, and Smalltalk programming interfaces. The OODB resource manager may also be accessed through application use of Open Blueprint Persistence Services. 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.

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

Persistence Services: The Persistence Service resource manager provides OO 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++ 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 Service 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, OO databases, and record files. The differences between these data stores can be hidden behind a common persistence interface.

The persistence service uses object designer-specified mappings when retrieving or storing objects that reside in or work with traditional data stores.

The persistence service allows an OO 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 Service is used in conjunction with other object services such as naming, transactions, and life cycle. The Persistence Service, for example, does not control transactions, but it cooperates with transaction management.

Storage Management: The Storage Management resource manager provides for protection and cost effective management of storage resources. These facilities are provided for 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 defines an application as the use of information technology to assist in the execution of a business process (see “Application Focus” on page 12). Distributed applications execute in or exploit multiple systems to accomplish their functions.

The Open Blueprint provides the structure and 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 distributed topology. The richness of these resource managers' functions allows the developer to easily incorporate distributed capability into the application. Application development tools enable the exploitation of these facilities to deliver integrated, interoperable business solutions. During development, tools create application components 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 defines the environment for distributed and/or 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, Conversation, and Messaging and Queuing), and the CORBA-compliant Object Request Broker using the Distributed System Object Model (DSOM).

Distributed solutions employ both distributed data and distributed logic across multiple locations. The Open Blueprint 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 parts 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, such as IBM's VisualAge, and rapid application development environments, such as IBM's VisualGen.

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 the distributed services of the Open Blueprint, 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 components that may execute on different systems. Each component 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 element added because of being distributed is the interaction between the application components.

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 components and analyzing the interactions among them

- Implementing a distributed application through generating application component skeletons from the design and allowing the developer to complete logic specifications, then building the components
- Evaluating a distributed application through distributed debuggers and performance recording and simulation
- Deploying a distributed application through enabling the tailoring of components 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 dvelopment environments such as the four described in “Development for Distribution” on page 59, 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 components which interface with resource managers of the Open Blueprint 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 allows for the manageability of resources across multi-vendor, open, enterprise-wide distributed systems. This is enabled through the systems management component structure. The structure is based on SystemView. SystemView defines the required management applications, management services and structure. SystemView and the Open Blueprint describe complementary aspects of capabilities needed to address 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 SystemView. 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 19 on page 63 shows the structure of this backplane. As depicted, this structure consists of four major elements. The four elements are:

- **Systems management applications** provide functions to interact directly with system administrators, systems programmers, and other personnel involved in the operation and administration of information systems. Systems management applications also provide the ability to automate the management tasks and processes. Systems management applications are categorized into disciplines as defined by SystemView:

  Business management supports tasks encompassing a wide range of enterprise business and administrative functions (e.g., budget planning, service agreement planning and control, software license management, security management, and asset management).

  Change management provides the controls and facilities to support the task of managing change in an information system environment (e.g., planning, scheduling, distributing, synchronizing, installing, activating, and monitoring changes to hardware and software resources).

  Configuration management consist of facilities needed to perform the tasks associated with planning, developing, and maintaining the operational properties and relationships of hardware and software resources (such as inventory control, configuration design, validation, environmental planning and creation).

  Operations management supports the tasks associated with the planning, distributing, evaluating, and controlling the workload in information systems (such as starting and stopping systems, subsystems and applications, receiving and responding to commands, setting policies regarding resources).

Performance management addresses the tasks for ensuring the effectiveness with which information systems deliver service to its users (such as monitoring resources and adjusting parameters with defined policies for achieving service level goals).

Problem management supports the tasks for managing problems and incidents from their detection through final resolution. (such as problem detection, analysis, recovery, resolution and tracking of faults, problems, or incidents occurring in the information system).

- **Systems management services** provide common services to resource managers and systems management applications unique to the needs of systems management. Through the use of these services and services provided by other resource managers, systems management applications will have an integrated appearance to the end user, share data among applications, and be shielded from the underlying technology used to access management information from remote systems. Some examples of systems management services are:

  Systems Management Presentation Services: provide user interface services required for the unique needs of systems management applications.

  Topology Display Services: used to display the relationships among managed resources.

  Autodiscovery Services: used to automatically discover resources in a network.

  Problem Management Services: used to work with problem task information (i.e., problem reports).

  Managed Resource Repository Services: provides a set of facilities for accessing information about managed resources.

  Management Protocol Services: extensions to the Open Blueprint Distributed System Services providing access to managed resources via the SNMP, CMIP, and SNA/MS management protocols.

  Encapsulation Services: provide object interfaces to selected procedural functions.

  Resource Correlation Services: provide the ability to correlate resources based on common attributes.

- **Agents** provide services to resource managers to facilitate interaction with systems management applications.
- **Managed resources** are resources owned by resource managers and are available for interrogation and control by systems management applications.

The systems management component structure also includes specific functional support that parallels resource manager structure. Two examples of resource manager structure are software distribution and software license management.

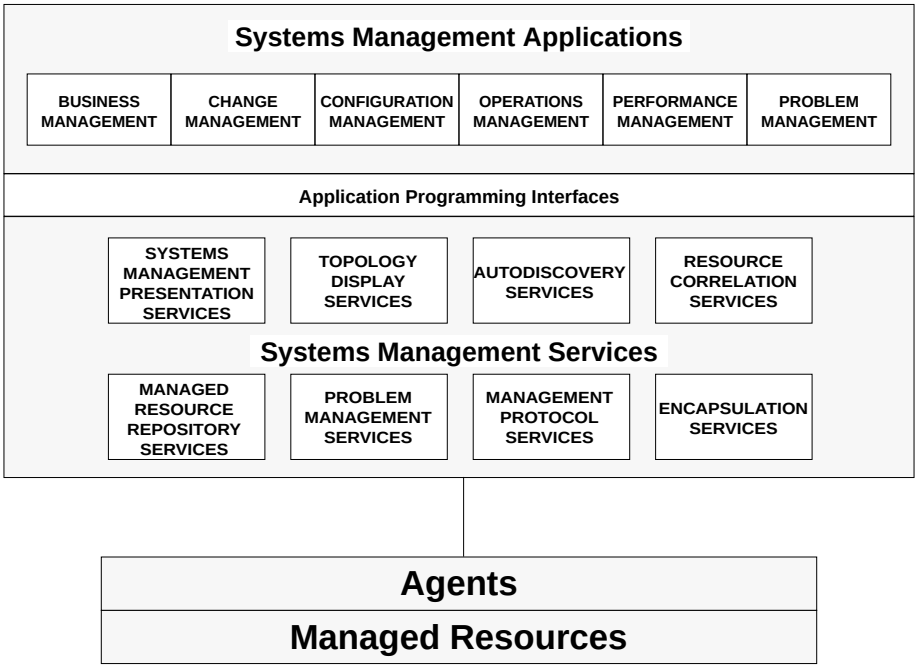

Figure 19. Systems Management Component Structure

Management Protocols

The management protocols supported encompass a broad range of **de jure** and **de facto** standards. These protocols support interaction between agents and systems management services allowing systems management applications access to managed resources. The key protocols are the Simple Network Management Protocol (SNMP), Common Management Information Protocol (CMIP), and Systems Network Architecture/Management Services (SNA/MS). The Open Blueprint Object Request Broker is used by objected-oriented systems management applications and managed resources.

Relationship to Resource Manager Structure

Resource managers are responsible for both the functional and management support aspects of their resources as described in “Resource Managers” on page 13. Systems management applications use APIs (i.e., XMP and SNMP APIs) to access the resource manager systems management support functions. These support functions use the systems management services as an important portion of resource manager systems management support.

Given that resource managers are distributed, some systems management services will be in the client part of the resource manager, some will be in the server part. Agent function typically resides in the server part. Communication between systems management services and agent function is typically through the standard systems management protocols.

Evolution of Systems Management Applications and Agents

Systems management software is evolving from procedural to object-based implementations. While this evolution is transparent to end users, special consideration is required in supporting resource managers and agents. Resource managers and agents deployed today using procedural technology must continue to work with new systems management applications implemented using object technology. Support for this evolution is provided through the use of encapsulation services and objected-oriented agents.

Manageability Needs

The systems management component structure is necessary but not sufficient for a manageable system. There are also requirements on the design of the resource managers. Resource managers use systems management services to allow systems management applications access to their managed resources. This includes using systems management services to:

- Provide notification of significant conditions to enable problem management,
- Provide vital product data for inventory and asset management,
- Respond to commands for operations management,
- Collect performance information for performance management,
- Use the software license management APIs for software license policy management, and
- Use local operating system facilities such as logs, traces, and storage dumps to facilitate problem management.

Management Standards

The Open Blueprint systems management supports **de jure** and **de facto** industry standards. The standards supported by the systems management component structure are

- The OSI X.700 Management Model
- The Simple Network Management Protocol (SNMP), Common Management Information Protocol (CMIP), and Systems Network Architecture/Management Services (SNA/MS) protocols
- The industry-standard APIs X/Open Management Protocol (XMP) and SNMP.

There are several emerging standards for systems management:

- POSIX System Administration services (1003.7 group)
- The Desktop Management Task Force (DMTF) Desktop Management Interface (DMI) APIs
- OSF/DME NetLS API for License Management
- X/Open System Management Object Services APIs

The systems management component structure will continue to evolve to support additional management protocols and APIs as they emerge.

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 memory, CPUs, or devices:

- 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 communications 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.

- Memory management

  Facilities for the management of memory resources. Distributed resource managers or servers must be able to access these resources where appropriate. The Open Blueprint Object Management Services utilize object-oriented APIs to memory management services.

- 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 activity can be started.

- Security context management

  Facilities for storage of and access to security credentials through the Generic Security Services API (GSSAPI).

- Multimedia system services

  Facilities for device management and data stream control/synchronization to support physical multimedia devices such as cameras, speakers, and displays.

- 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).

- Language Environment/POSIX

  The 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.

  These facilities will be extended in scope and will converge over time with POSIX initiatives. That is, local operating system resource managers will adhere to the POSIX 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.

Appendix A. Standards

The following table shows some principal industry interfaces and protocols in the Open Blueprint. It is based on formal standards that have achieved widespread industry acceptance.

The Open Blueprint also includes several *de facto* standards, either in the absence of a formal standard, or to achieve interoperability with existing installations.

References for ISO/IEC, IETF, and ITU-T are to Standards Number. References for X/Open are to International Standard Book Number (ISBN). You can get information on the company-contributed interfaces and protocols from the contributing company.

IBM Interfaces	**Document Order Number**
CICS API	SC33-1007
DL/I	SC26-8015
IMS TM API	SC26-8017
MQI	SC33-0850
OS/2 PM API	ISBN 1-56529-155-7

IBM Protocols	**Document Order Number**
APPC	SC31-6808
APPN	SC30-3422
DDM	SC21-9526
DL/I	SC26-8015
DRDA	SC26-4651
IPDS	S544-3417
MPTN	GC31-7073/4
SNA Sync Point	SC31-8134
SNA/MS	SC30-3346
SNA Formats	GA27-3136

Common Transport Semantics

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

Transport Network

Interfaces	Protocols	Organization	Reference
XTI		X/Open	ISBN 1-872630-29-4
Berkeley Sockets		Berkeley	4.3 BSD Document
	APPN	APPN Implementors Workshop	IBM - *
	TCP/IP	IETF	791, 793, 768
	OSI	ISO/IEC	8072/3, 8602, 8473
	NetBIOS	IBM	*
	IPX	Novell	*
	Appletalk	Apple	*

Signalling and Control

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 3.1

* The contributing companies may be contacted for more details. The IBM document numbers are on page 67.

Conversational Communication

Interfaces	Protocols	Organization	Reference
CPIC		X/Open	ISBN 1-85912 057-1
	APPC	IBM	*
	OSI TP	ISO/IEC	10026

RPC Communication

Interfaces	Protocols	Organization	Reference
DCE RPC	DCE RPC	X/Open	ISBN 1-85912-0679

Messaging and Queuing

Interfaces	Protocols	Organization	Reference
MQI		IBM	*

Object Management

Interfaces	Protocols	Organization	Reference
CORBA		OMG	93-12-43
Common Object Services		OMG	94-1-1
Lifecycle Services		OMG	93-7-4
Externalization Services		OMG	94-9-15
Identity		OMG	94-5-5
	IIOP, DCE-CIOP	OMG	95-3-10

Directory and Naming

Interfaces	Protocols	Organization	Reference
XDS		X/Open	ISBN 1-872630-18-9
XOM		X/Open	ISBN 1-85912-0784
Directory Service	Cell Directory Service	X/Open	ISBN 1-872630-17-0
	X.500	ISO/IEC	9594 (88 Standard)
	Domain Name Service	IETF	1034,1035
Naming Service		OMG	93-5-2

* The contributing companies may be contacted for more details. The IBM document numbers are on page 67.

Security - User Registry

Interfaces	Protocols	Organization	Reference
DCE SEC_RGY	DCE Registry	OSF	DCE Security Specification

Security - Authentication

Interfaces	Protocols	Organization	Reference
GSSAPI		IETF/OSF	DCE Security Specification
	Kerberos	MIT/OSF	Kerberos draft

Security - Access Control

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

Time

Interfaces	Protocols	Organization	Reference
DCE DTS	DCE DTS	X/Open	ISBN 1-85912-0415

Transaction Manager

Interfaces	Protocols	Organization	Reference
Transactional C	Encina	Transarc	*
TX		X/Open	ISBN 1-872630-65-0
XA		X/Open	ISBN 1-872630-24-3
	OSI TP	ISO/IEC	10026
CICS API		IBM	*
IMS TM API		IBM	*
	SNA Sync Point	IBM	*
Transaction Service		OMG	94-8-4

* The contributing companies may be contacted for more details. The IBM document numbers are on page 67.

User Interface

Interfaces	Protocols	Organization	Reference
OS/2 PM		IBM	*
Motif		OSF	MOTIF Specification
Windows		Microsoft	*
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

Distributed Print

Interfaces	Protocols	Organization	Reference
Distributed Print API		IEEE	POSIX 1387.4/d7
	DPA	ISO/IEC	10175-3
	lpr/lpd	IETF	1179
	SMB	X/Open	ISBN 1-872630-45-6
	NCP	Novell	*

Print Data Streams

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

Distributed 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	*

Relational Database

Interfaces	Protocols	Organization	Reference
SQL (1992E)		ISO/IEC	ISBN 1-872630-63-4
SQL CLI		X/Open	9075
	DRDA	IBM	*
ODBC		Microsoft	*

* The contributing companies may be contacted for more details. The IBM document numbers are on page 67.

Hierarchical Database

Interfaces	Protocols	Organization	Reference
DL/I		IBM	*

Persistence Services

Interfaces	Protocols	Organization	Reference
POS		OMG	94-1-1 94-10-7

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 Services		OMG	93-7-3

Mail

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

* The contributing companies may be contacted for more details. The IBM document numbers are on page 67.

Telephony

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

Systems Management

Interfaces	Protocols	Organization	Reference
	SNA/MS	IBM	*
XMP		X/Open	ISBN 1-872630-32-4
	CMIP	ITU-T/ISO/IEC	X.711
SNMP	SNMP	IETF	1157
DMI		DMTF	*

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 67.

Bibliography

The Open Blueprint Publications

Table 1. Open Blueprint Publications

Open Blueprint:	
Introduction to the Open Blueprint	G326-0395
Technical Overview	GC23-3808
Open Blueprint: Technical Reference Library (This publication includes Open Blueprint component description papers as well as the *Introduction to the Open Blueprint* and *Open Blueprint Technical Overview*.)	SBOF-8702 SK2T-2478

Related Publications

Table 2. Related IBM Publications

IBM Security Architecture	SC28-8135
Multimedia Distributed Computing - IBM's Direction for Multimedia Distributed Systems	G229-7340

Other Publications

Table 3. Related Books That Are Not Published by IBM

X/Open Company Ltd., *Distributed Computing Services Framework*, ISBN 1-872630-64-2, 1992.
Gray, *Open Systems and IBM: Integration and Convergence*, London, McGraw-Hill, 1993.

Readers' Comments — We'd Like to Hear from You

Open Blueprint
Technical Overview

Publication No. GC23-3808-01

If you especially like or dislike anything about this book, please tell us what you think. Feel free to comment on specific errors or omissions, accuracy, organization, subject matter, or completeness.

To request additional publications or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer. Use this form for comments about the information in this book and how the book presents the information.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

To send us your comments:

- By mail, use this form.

  If you are mailing this form from a country other than the United States, you can give it to the local IBM branch office or IBM representative for postage-paid mailing.

- By FAX, use this number:

  FAX (United States & Canada): 914+432-9405
  FAX (Other Countries):
  Your International Access Code +1+914+432-9405

- Electronically, use any of the following network addresses:

  IBMLink (United States customers only): KGNVMC(D58PUBS)
  IBM Mail Exchange: USIB2NZL at IBMMAIL
  Internet: d58pubs@vnet.ibm.com

If you would like a reply, be sure to include your name, address, telephone number, or FAX number.

Make sure to include the following in your comment or note:

- Title and order number of this book
- Page number or topic related to your comment

Name _____ Address _____

Company or Organization _____

Phone No. _____

Readers' Comments — We'd Like to Hear from You
GC23-3808-01

Cut or Fold
Along Line

IBM®

Fold and Tape | **Please do not staple** | Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
Department 55JA Building 708-2, Mail Station P384
522 South Road
Poughkeepsie, N.Y. 12601-5400

Fold and Tape | **Please do not staple** | Fold and Tape

Cut or Fold
Along Line

Printed in U.S.A.

GC23-3808-01