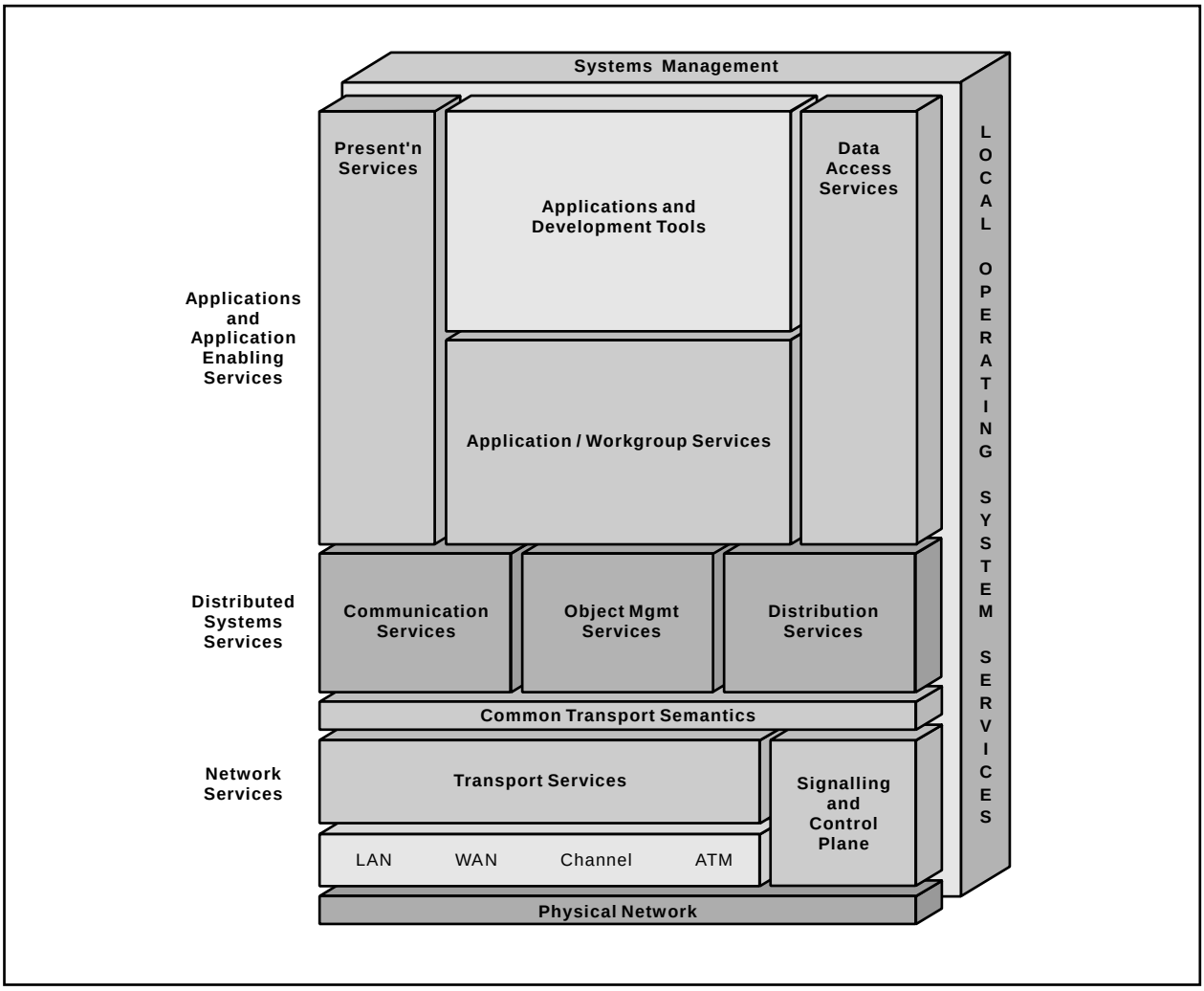
Open Blueprint

File Resource Manager



GC23-3917-01

Open Blueprint

File Resource Manager

GC23-3917-01

About This Paper

Open, distributed computing of all forms, including client/server and network computing, is the model that is driving the rapid evolution of information technology today. The Open Blueprint structure is IBM's industry-leading architectural framework for distributed computing in a multivendor, heterogeneous environment. This paper describes the File resource manager component of the Open Blueprint and its relationships with other Open Blueprint components.

The Open Blueprint structure continues to accommodate advances in technology and incorporate emerging standards and protocols as information technology needs and capabilities evolve. For example, the structure now incorporates digital library, object-oriented and mobile technologies, and support for internet-enabled applications. Thus, this document is a snapshot at a particular point in time. The Open Blueprint structure will continue to evolve as new technologies emerge.

This paper is one in a series of papers available in the *Open Blueprint Technical Reference Library* collection, SBOF-8702 (hardcopy) or SK2T-2478 (CD-ROM). The intent of this technical library is to provide detailed information about each Open Blueprint component. The authors of these papers are the developers and designers directly responsible for the components, so you might observe differences in style, scope, and format between this paper and others.

Readers who are less familiar with a particular component can refer to the referenced materials to gain basic background knowledge not included in the papers. For a general technical overview of the Open Blueprint, see the *Open Blueprint Technical Overview*, GC23-3808.

Who Should Read This Paper

This paper is intended for audiences requiring technical detail about the File Resource Manager in the Open Blueprint. These include:

- Customers who are planning technology or architecture investments
- Software vendors who are developing products to interoperate with other products that support the Open Blueprint
- Consultants and service providers who offer integration services to customers

Contents

Figures

File Resource Manager

Introduction

The File resource manager component of IBM's Open Blueprint refers to application enabling services for managing file resources in an open, distributed environment. Specifically, the File resource manager is one of the data access services of the Open Blueprint.

The File resource manager is an application enabler that utilizes local system services for physical storage of files, and utilizes elements of the Distributed Services and Network Services layers of the Open Blueprint to support global access to files. The positioning of the Data Access Services in the Open Blueprint is illustrated on the cover.

File resource managers are used to store and retrieve persistent data objects on direct access storage devices. They support the basic data storage requirements of the system and its applications. This includes file Directory services for recording file attributes and where the files are stored that are integrated with the Open Blueprint Directory services for locating resources in distributed environments.

The Data Access Services in the Open Blueprint includes database and storage management services, as well as File resource manager services. The positioning of the File resource manager in the context of the Data Access Services is illustrated in Figure 1.

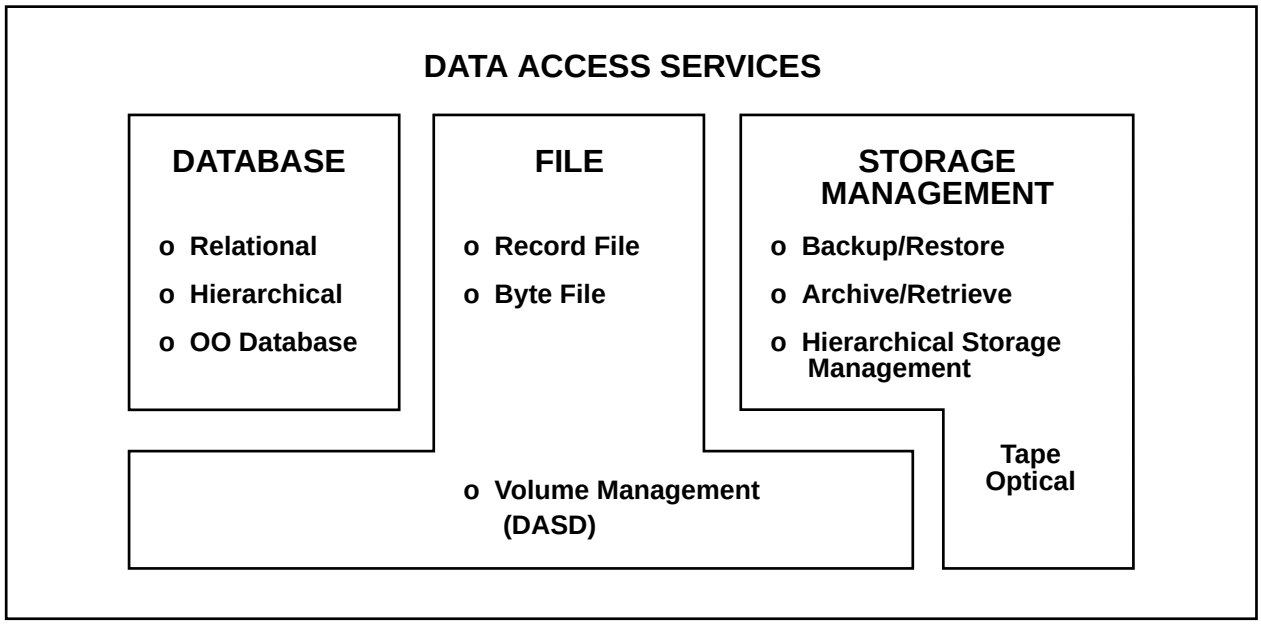


Figure 1. File Resource Manager in Data Access Services

The File resource manager provides the computing environment with the most basic data services for persistent storage of data. At the lowest level, this is volume management for placement of data on direct access storage devices. Above that layer are either byte file or record file services.

Database services would typically be built on top of either the volume management services or byte file services of the File resource manager. The Storage Management services would provide the listed services for file and database resource managers. This paper covers the topics listed under FILE in Figure 1. Discussion of the topics listed under Database and Storage Management are discussed in separate Open Blueprint papers.

File System Requirements and Background

File systems provide the basic data handling functions of any operating system and application-enabling environment. To a large extent, the capabilities and characteristics of an operating system are defined by the file system support. System characteristics, such as performance, scalability, security and reliability are due in large part to the corresponding characteristics of its file system. Indeed, the naming conventions of a system, including program and command names derive from naming conventions of the file system.

Similarly, the types of applications supported by an operating system are heavily influenced by the file-handling support in the system. Commercial transaction processing, multimedia and open, portable applications all imply requirements on the underlying file handling capability of the system.

By the same token, the capabilities and characteristics of a distributed computing environment are also heavily dependent upon the distributed file handling capability. This paper explores the challenges facing file system support in open, distributed environments in the context of many of the challenges faced by open distributed systems in general. Based on these challenges, a direction is identified that addresses the role of file systems in distributed system solutions.

Key challenges that need to be addressed include:

Data addressing
: Locating files that may be distributed across multiple systems and servers needs to be addressed in an intuitive and consistent fashion.

Data location independence
: Addressing of data should be independent of the data's location. That is, to allow configuration flexibility, the location of files should not be carried as part of the file naming mechanism.

User location independence
: Access to data should not assume a specific system or workstation location for the user. A user should be able to obtain access to data (as authorized) from multiple locations.

Application portability
: Standard application file interfaces need to be supported to accommodate maximum flexibility for deploying applications on systems in the distributed computing environment.

Coexistence of Multiple File Implementations
: The File resource manager must encompass and support data and file systems that exist today. It is not realistic to expect customers to migrate their data to participate in the distributed computing environment.

Multiple Protocol Support
: Given there are multiple distributed file access protocols that are commonly used in the industry, the File resource manager needs to be able to support multiple protocols.

Performance
: The performance of the File resource manager is critical to the performance of both the local system and distributed system processing. In distributed environments, techniques are required that allow access to remote data to approximate access to local data.

Scalability
: The File resource manager needs to be able to scale from stand-alone environments to large enterprise environments that support thousands of users. This includes support for parallel system environments.

Reliability
: The File resource manager needs to provide reliable support for file data, in all environments. This includes stand-alone and client/server environments, as well as large enterprise environments.

Availability

Data availability issues need to be addressed, particularly in distributed environments. The File resource manager needs to tolerate and isolate failure conditions.

Ease of Administration

Given the diversity of file systems and the distribution of data across multiple nodes in a network, there is a need for a consistent set of tools for managing data.

Concepts

Figure 2 illustrates the basic objects managed by the File resource manager.

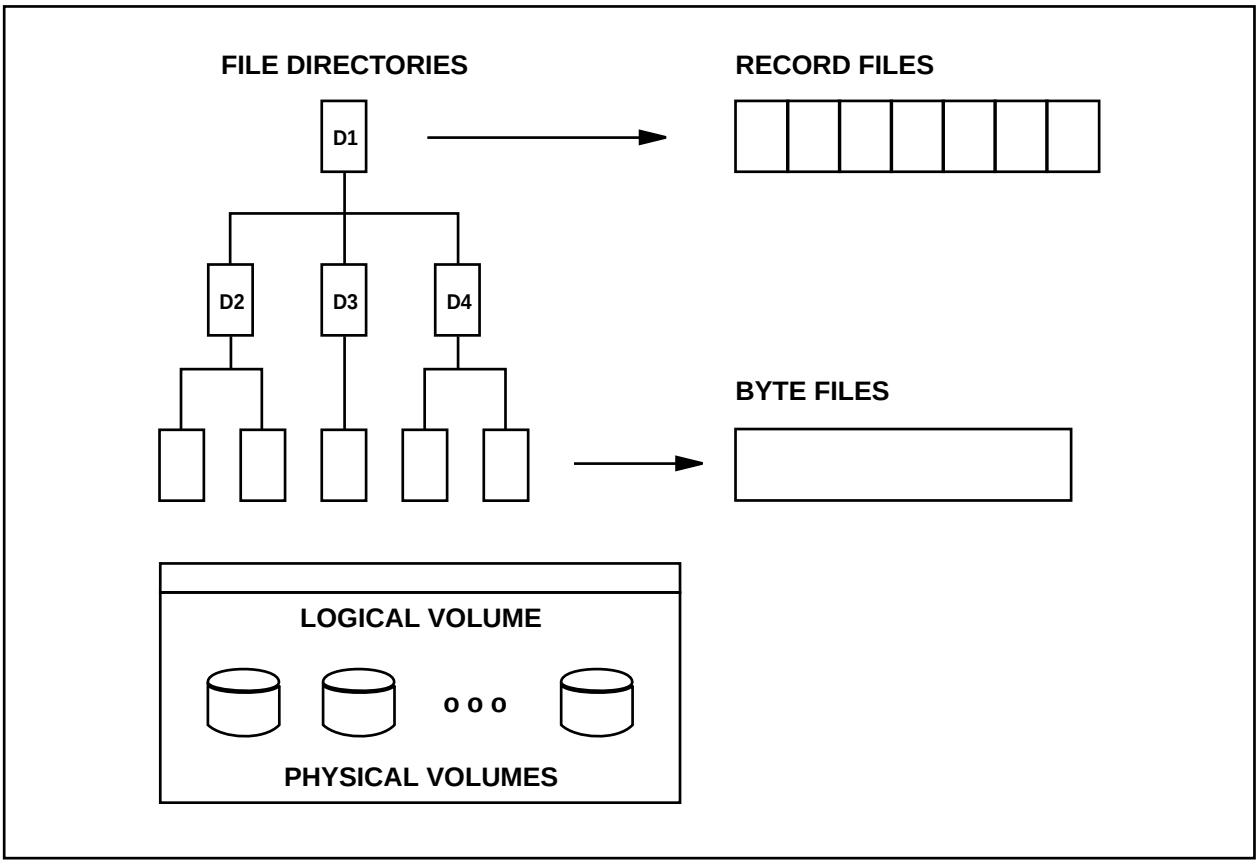


Figure 2. File Storage Objects

Files

A file is a named data object that is stored on persistent storage media (for example, DASD), and is accessed (read and written) using file operators. A file is a common construct for storing application data for future use.

Files can be one of two types:

- Byte stream files or
- Record files.

Byte Stream Files

A byte stream file is a file that is operated on by byte-level read and write operations. The file appears to applications as one long stream of bytes. Read and write operations are performed on any application-specified range of bytes in the stream.

Record Files

A record file is a file that is operated on by record-oriented read and write operations. The file appears to applications as a set of records, each of which can contain multiple bytes. Read and write operations are performed on any application-specified record (or records) in the file, but not on any arbitrary range of bytes in the file.

File Directories

A file directory is the mechanism for recording file location and finding files in storage. When used in conjunction with the distributed directory service of the Open Blueprint, they can be used to locate files in a network.

File directories can be *flat* or *hierarchical*. A flat directory is a single directory (no subdirectories) of files that provides a single index into all files stored under one File resource manager instance (file system). A hierarchical directory provides a nesting of directories as a means of indexing files in a tree structure composed of files and lower-level directories (called subdirectories).

Logical Volume

A logical volume is a named reference to physical storage that has been reserved for storing data. A logical volume is composed of allocated space on real physical volumes. A logical volume is a logical abstraction of a physical volume, and appears to the data manager as a physical volume. Files and the file directories that index them are stored in logical volumes. Logical volumes are also used to hold database data.

Physical Volume

A single DASD device media (for example, one harddisk and one floppy). When data is written to a logical volume, logical storage addresses are translated to real device locations on physical volumes.

In the context of the objects managed by File resource manager, some of the key File resource management concepts are illustrated in Figure 3.

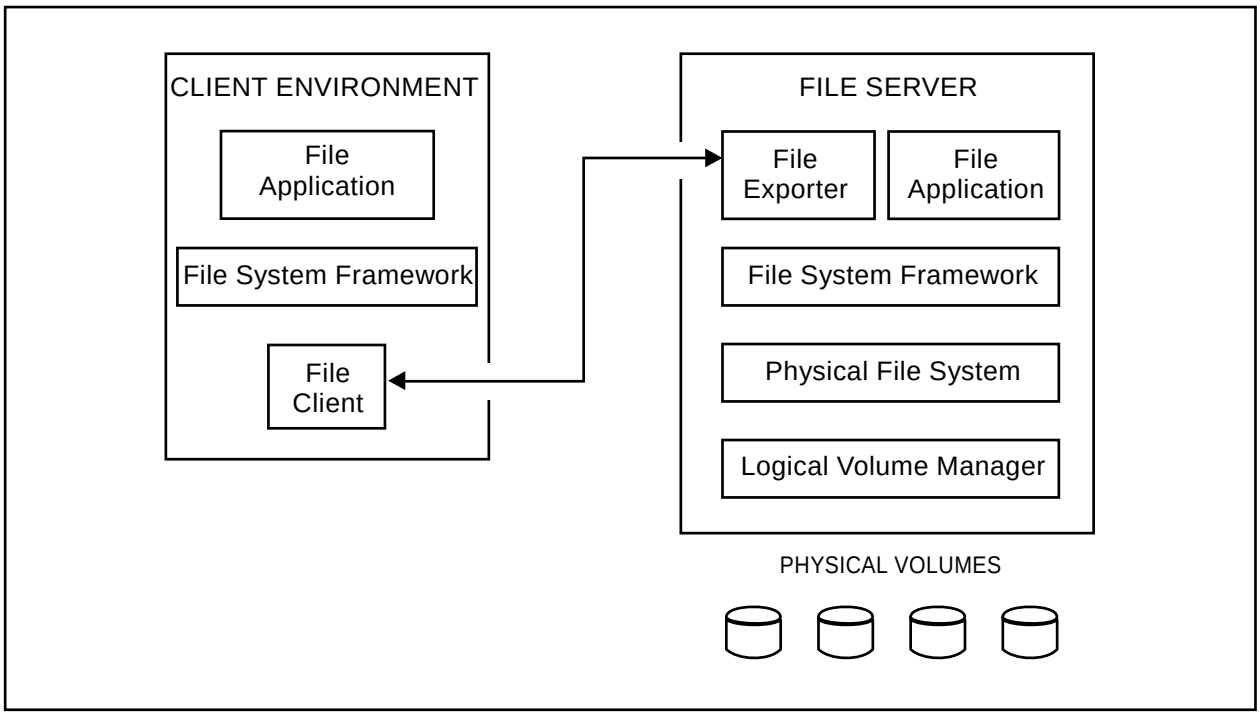


Figure 3. File Resource Manager Key Concepts

The key elements illustrated in Figure 3 are described below:

File System Framework

The File System Framework provides a consistent, robust, easy-to-use set of file services to applications and subsystems for file storage and retrieval. The File System Framework also provides access to any number of physical file system implementations.

File System Client

A file system client refers to code that operates in an application environment and translates application file requests to network file access protocol, to provide access to a remote file system in the network (or LAN).

File Exporter

A file exporter presents a network interface to a file system for the purpose of accessing file data across a network (or LAN). File exporters run on a local file system (or file system framework) and support specific network protocols for file access. There are several network file access protocols common in the industry. A file exporter would translate one such network protocol to local file system operations.

File System Server

A file system server refers to code that runs on the system that holds the file data, and exports network file access to file system clients using one or more network file access protocols. That is, a file system server includes support for one or more file exporters for remote access to the file systems it manages.

Logical Volume Manager (LVM)

A service that exports a device driver interface, but manages multiple physical volumes as one or more logical volumes. An LVM also exports administrative services for configuring real (physical) volumes into logical volumes.

Physical File System (PFS)

An implementation of a specific local file system.

File Functions

In the context of IBM's Open Blueprint, the File resource manager refers to components that provide the following services:

File Services

Application services for reading and writing file data, and utilities that operate on files (for example, copying files).

File System Services

Services and utilities that operate on whole file systems. This includes services for creating and altering file systems.

FS Directory Services

Services that operate on file system directories. This includes services for creating directories, renaming files and managing file attributes.

Volume Services

Services for managing the direct access storage for file systems. This includes services for allocating and expanding storage and defining special storage attributes such as mirroring or striping.

File Client Support

This function runs under application file services interfaces to communicate file requests to remote systems.

File Exporter Support

This function runs on a local File resource manager and handles file requests from remote (client) users.

File Resource Management Structure

Although implementations of the File resource manager can vary across operating system environments, the general component structure that is followed is illustrated in Figure 4.

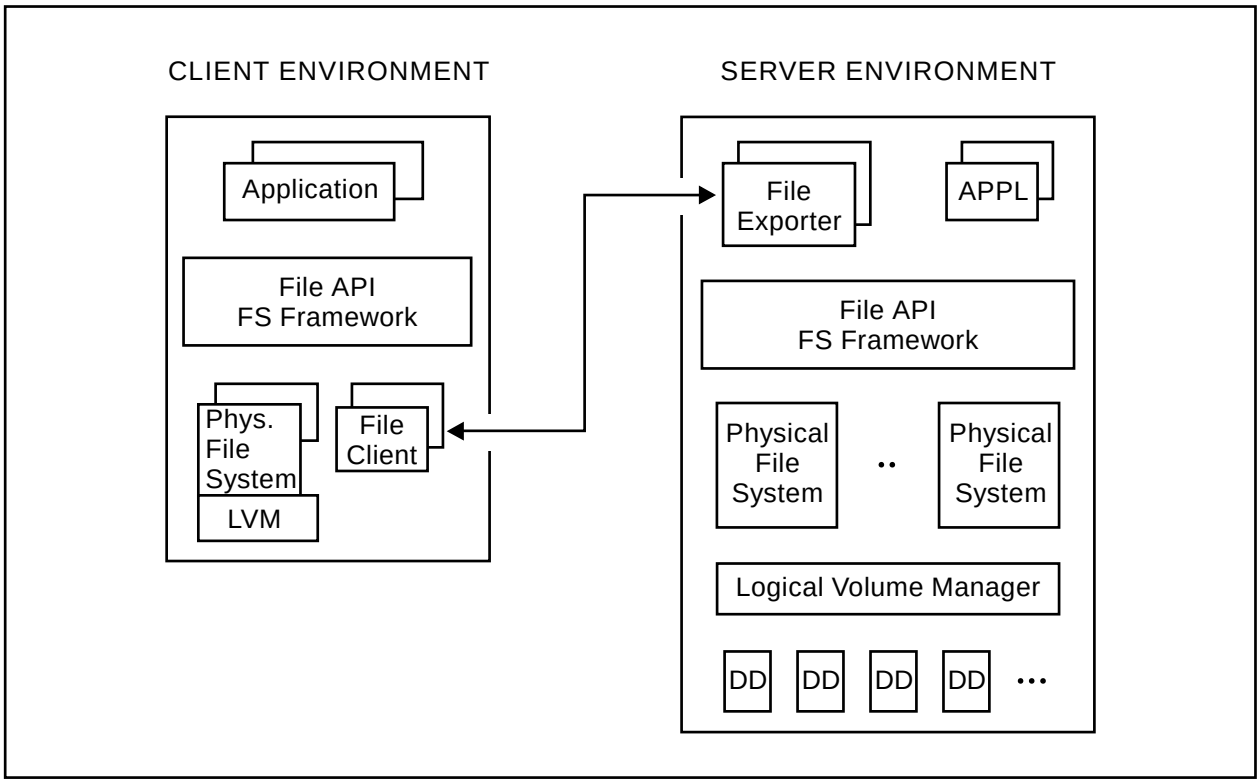


Figure 4. File Resource Manager Components

File System Frameworks

Supply file APIs to applications (and file exporters) for file services, file system services and file system directory services, and route file system requests to the appropriate physical file system or file client. While most of the functions are actually implemented by the physical file systems underneath the framework, the framework can perform certain key functions, such as resolving a file or directory reference to the appropriate physical file system or file system client. To a limited extent, it can also mask differences among physical file systems underneath it.

The file system framework plays a key role in supporting location independence and name-based file addressing. Since it provides the API support, it is also the key to application portability.

One or More Physical File Systems

(sometimes referred to as *local file systems*) Specific implementations of file access, manipulation and storage. By supporting multiple physical file systems, the File resource manager can provide support for specific file handling needs (for example, CD-ROM support and legacy data support).

A PFS provides a specific file system implementation. As a result, it is the component where specific file system requirements are addressed. Many of the performance, scalability, reliability and availability requirements are addressed by specific physical file system implementations tuned

to specific requirements of environments. For example, journaling for reliability or parallelism would typically be addressed at the physical file system level. Another key example would be support for legacy data or support for standard file system data formats (such as FAT or CD-ROM formats).

Support for multimedia serving would typically call for a physical file system that has special characteristics in support of guaranteed delivery of data. This could include striping and caching, or other techniques for ensuring rapid access to data that is usually read.

The physical file system layer may also be used to support adapting the services of the framework APIs to an existing legacy file system. That is, instead of a real physical file system, an adapter is invoked from the framework to map file requests to those of an existing file system.

In general, the structure which allows multiple physical file system implementations under the framework enables file system implementations that are tailored to specific applications or customer environments.

Logical Volume Manager(s)

Manage direct access storage for file systems (and database managers). The logical volume manager layer masks specific physical storage considerations from physical file system (or database manager) implementations.

The Logical Volume Manager provides volume services to multiple data managers (database managers, as well as physical file systems). As such, it is a natural place to provide functions that are commonly needed across multiple data managers. Such functions include spanning of physical volumes, mirroring and even striping.

File Exporters

Support access to local file systems from network users (file clients). There can be multiple file exporters to support the variety of network protocols common in the industry. File server environments would typically support multiple file exporters to accommodate a variety of client systems.

File exporters are key to supporting a heterogeneous distributed computing environment. A file exporter is responsible for translating file requests, as expressed by the client, to framework file services. There would be a file exporter for each of the client protocols to be supported by the File resource manager.

Note: In addition to supporting standard file access protocols, file exporters may also be developed to support special distributed file processing requirements. An example of this can be found in support for multimedia applications. To meet requirements for guaranteed delivery and special multimedia functions, extensions to standard file protocols may be required.

File Clients

Provide remote file access services to file applications in the client system environment. Under the file system framework, multiple file clients may be implemented to support access to various file servers. Generally speaking, client environments would tend to support only one remote file access protocol. In order for a client system to be able to access files in a file server, it needs only to have a file client that supports a file protocol that matches **one** of the protocols supported by the file exporters in the server environment.

Note: File clients fit underneath the file system framework, much like a physical file system. In effect, the file system client acts as a surrogate for the remote file system.

The structure shown in Figure 4 on page 6 is a general structure that supports a variety of file access protocols and physical file system implementations. Each system implementation may have some variations of the structure shown.

Furthermore, there are certain optimizations of the structure that can be done for “local” data access and homogeneous distributed access. Such optimizations do not, in general, invalidate the more general structure for supporting “non-optimized” file systems or file protocols.

Byte File Structure

The general component structure for byte file support is illustrated in Figure 5.

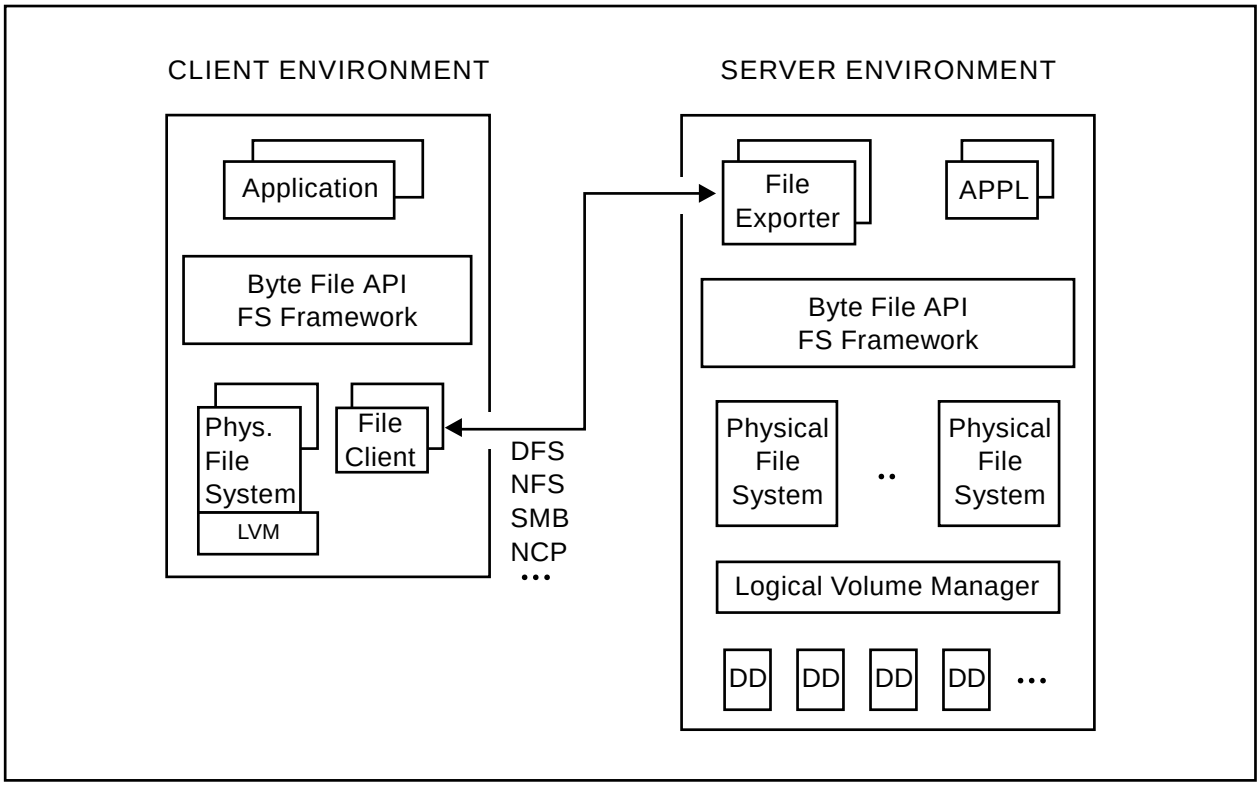


Figure 5. Byte File Resource Manager Components

File system frameworks

For byte files, the file system framework supports a variety of file exporters and a variety of physical file systems and file clients. In addition, the framework may support file functions that are unique to local system applications. The framework supports the standard application interface (POSIX 1003.1).

One or More Physical File Systems

For byte files, multiple PFS implementations would typically be supported, depending on the system environment. For example, for workstation environments there could be support for the FAT file system, HPFS, CD-ROM and/or a journaling file system (such as the OSF DCE local file system). Specific PFS support would typically be different for different system environments, based on the specific requirements of the system.

Logical Volume Manager(s)

The logical volume manager layer of the File resource manager structure is not data manager-specific. Thus, it has no functions that are specific to byte files. However, here it should be noted that legacy PFSs (for example, CD-ROM support) may implement volume management functions as part of the PFS. Similarly, certain special case PFSs (for example, multimedia file systems) may also incorporate volume management functions as part of the PFS.

File Exporters

For byte files, there are a variety of file access protocols prevalent in the industry. As a result, systems that act as file servers would typically support multiple file exporters (one for each of the industry common byte file protocols). This will be discussed in more detail in “Distributed File Access” on page 13.

File Clients

For byte files, a client system would typically be configured with a limited number of file client types. However, the structure allows a system to be configured with multiple file clients. A single user workstation would typically be configured with one file client type (for example, NFS or DFS). A multi-user system or a gateway server system may well be configured with multiple client types.

Record File Structure

The general component structure for record file support is illustrated in Figure 6.

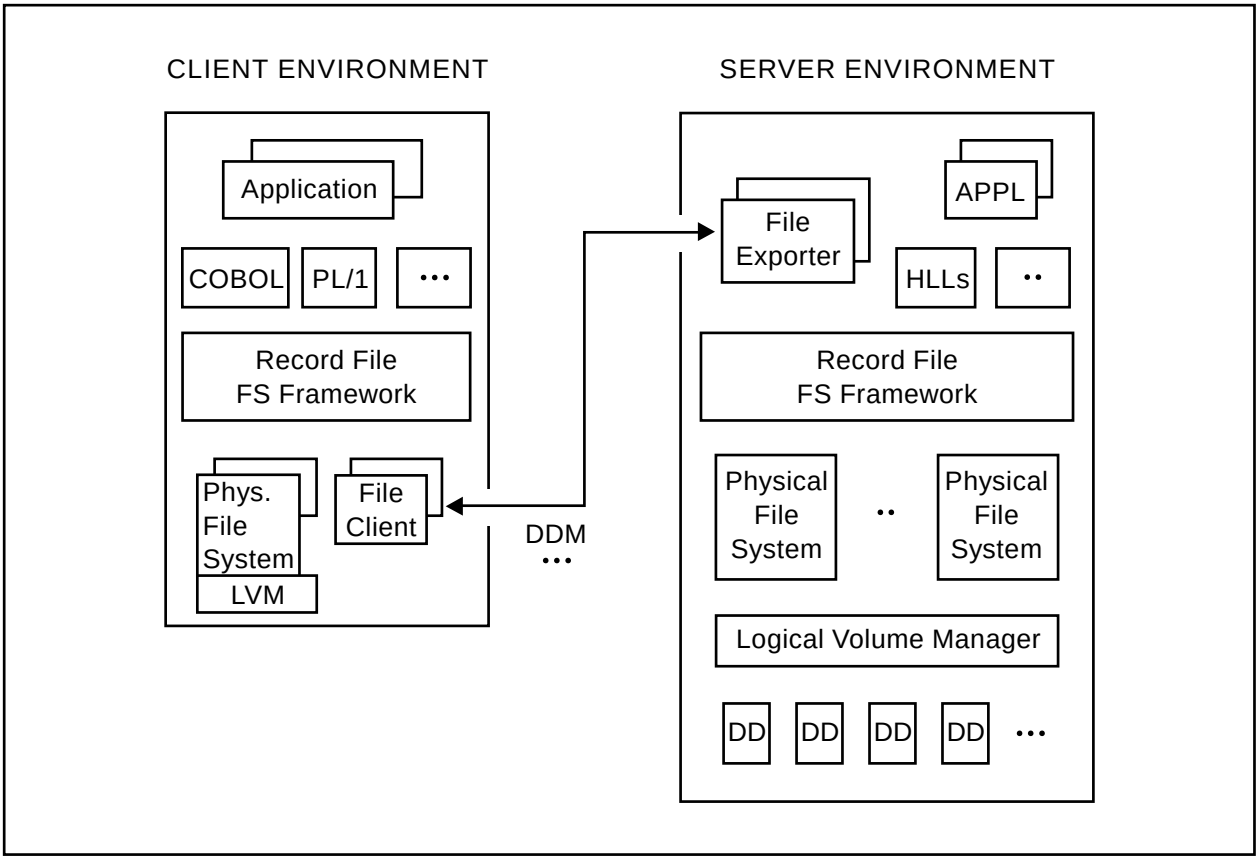


Figure 6. Record File Resource Manager Components

File system frameworks

For record files, the file system framework is capable of supporting multiple file exporters and multiple physical file systems and file clients. Also, the framework may support file functions that are unique to local system applications. Standard applications interfaces, however, are not directly implemented by the framework. Standards for record file operations are defined by high level languages, and direct support for the standards would be through the runtimes of those programming languages.

The record file services provided by record file frameworks vary by system implementation. Thus, the interfaces are system programming interfaces.

One or More Physical File Systems

For record files, multiple PFS implementations can exist. The record file systems supported vary by system platform. In workstation environments, where record file support is typically not provided as part of the base system offerings, vendors of record file systems can plug into the framework and pick up HLL support for their file system.

Logical Volume Manager(s)

The logical volume manager layer of the File resource manager structure is not data manager-specific. Thus, it has no functions that are specific to record files. However, here it should be noted that legacy PFSs (for example, VSAM support) may implement volume management functions as part of the PFS.

File Exporters

For record files, the Open Blueprint supports Distributed Data Management (DDM) protocol for "standard" access to local record file systems from network users (file clients). However, there are a number of "private" and de facto standard protocols that are available in the market. Thus, the architecture for multiple file exporters applies for record data to accommodate implementation of exporters for those protocols.

File Clients

For record files, the Open Blueprint supports the DDM protocol for client access to remote record files. However, as with file exporter support, the architecture accommodates driving other record file protocols from the record file system framework.

There can be a number of variations involved in the record file system support. This is particularly true in the area of transaction processing support for file data. While most of the File resource manager structural concepts apply, the actual component structure in any implementation can contain a number of variations.

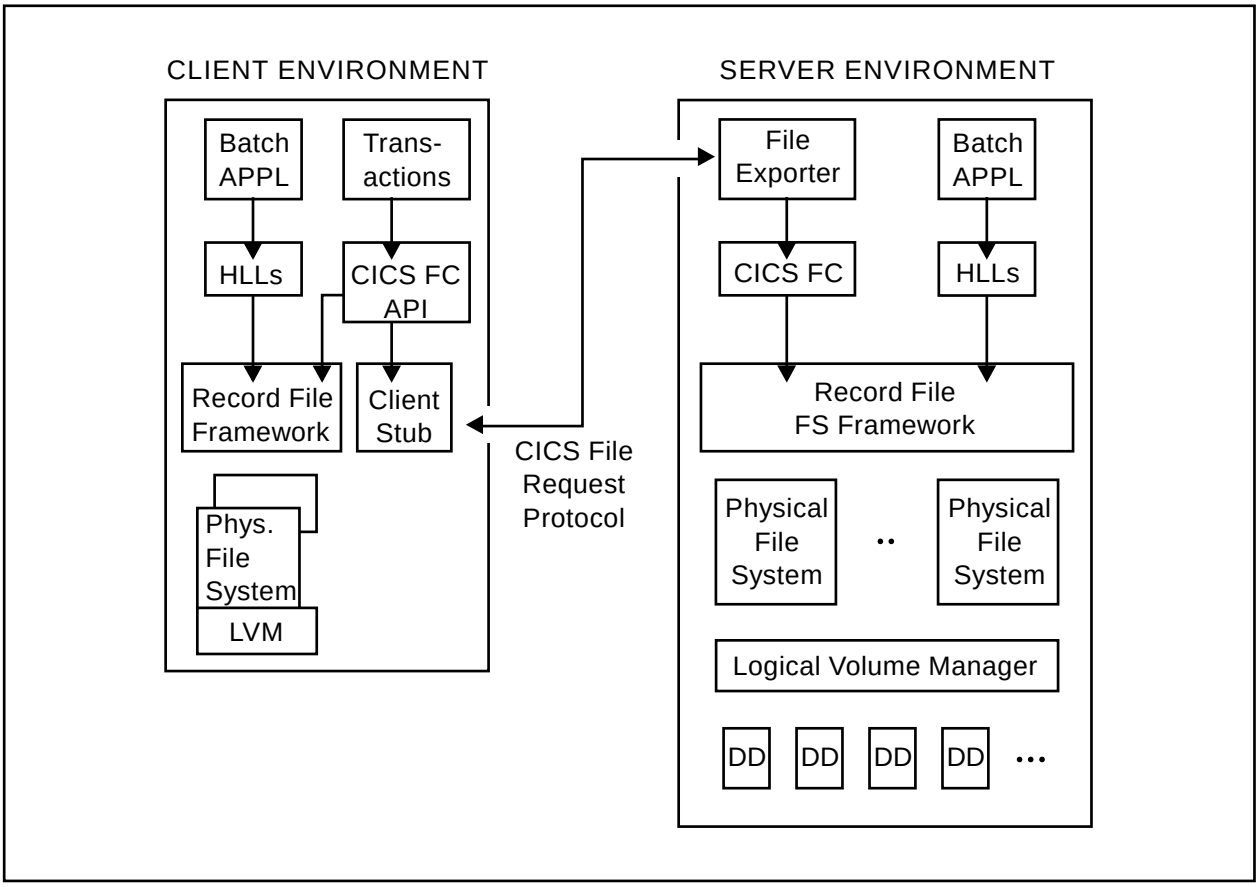


Figure 7. CICS Transactional Record File Support Components

For example, Figure 7 illustrates a variation of the record file support for CICS-based transaction environments. In this structure, many of the elements of the general structure for record file access are incorporated into CICS components. Yet the CICS components coexist with the general, non-transaction components. Some of the key elements illustrated in Figure 7 are:

CICS FC API

In the CICS environments, the CICS File Control API serves the role of the record file system framework. Exporters can be built on top of it, and the API can be mapped to multiple physical record file systems. In addition, the CICS FC API can route file requests to a CICS client stub for access to remote files.

The CICS FC API also provides an application programming interface that is available on a wide variety of platforms.

Client Stub

The CICS Client Stub provides the 'file client' support for accessing remote transaction file data using the CICS file request protocol.

File Exporter

In server environments, CICS provides a file exporter for catching and handling CICS File Requests. Like the general structure, the file exporter uses the framework on the server system (that is, CICS File Control) for accessing the physical record file system(s) on the server.

The CICS components provide the services and functions of many of the File resource manager components. But they coexist with the non-transactional record file components to provide a level of sharing between transaction and non-transaction applications. The level of sharing will vary depending on the locking protocols supported by the underlying physical record file system being used.

One of the reasons for variations in transactional record file support is the need to introduce transaction processing and interaction with the Open Blueprint Transaction Manager in the flow of file access. Non-transactional record services do not use the Transaction Manager components of the Open Blueprint.

File Programming Interfaces

The File resource manager of the Open Blueprint supports industry-standard file interfaces for procedural application programming services. In addition, there is a service provider interface between the file system framework and the physical file systems (or file clients). The service provider interfaces vary by specific system environment. Interfaces to the Logical Volume Manager function of the File resource manager are system programming interfaces that vary by system.

In addition to procedural interfaces, File resource managers provide support for object-oriented access to file data.

File Data Structures: File resource manager services are based on the data structures desired by the application (or the applicable resource manager). As a result, the services vary depending on whether the data structure being used is a byte file, record file or logical volume.

Byte File Data

The byte file structure is simply a named stream of bytes. Access to data in a byte file is by byte offset into the stream of bytes, and applications typically operate on a range of bytes in the file.

Record File Data

Record-Oriented file systems provide a variety of record organizations and methods for accessing the records they store. The choice of a record organization is determined by some compromise of performance, and the way the data is used by the application. The record organizations supported are:

Keyed

Data is accessed by a key field or index in the data.

Entry Sequenced

Data is accessed in the order it was entered.

Relative Record

Data is accessed by a record number.

Volume Data

Volume services operate on blocks of data storage allocated to a specific storage construct (referred to as a Volume in this paper). At the volume level, there is no notion of specific files, just blocks of storage that can be read or written. A block of storage may contain directory information, file data or database data.

Procedural File APIs: The APIs supported by the Open Blueprint are the basis for supporting application portability requirements for file applications. The APIs supported by the File resource manager of the Open Blueprint are the following:

Byte File API

Posix 1003.1 services for file system access.

Record File APIs

The record file interfaces supported by standard languages that provide record file functions (that is, COBOL, and PL/1).

Interfaces between language runtime routines and record file systems are system programming interfaces and vary across systems.

In addition to standard record interfaces defined by high-level languages, other application-level record interfaces can be found in the industry. Two of note are the CICS File Control interface, which is available on a number of platforms, and the Encina Structured File System (SFS) programming interface. These interfaces feature transactional support through their interface services. Other facilities that provide record services include Btrieve and LANDP.

Note: Volume services are typically supported as system programming interfaces for use by file systems or database managers. They are not intended for general application use. At this time, there are no standards for volume services interfaces. Volume services vary by system platform, and they are most commonly referred to as Device Driver interfaces.

File Framework SPIs: In addition to the APIs supplied to applications, the file framework provides a service provider interface (SPI) for supporting multiple physical file system implementations, and file clients under the framework. This mechanism allows applications written to the file API to gain transparent access to any file system implementation and any protocol-specific file client.

The interface between the framework and physical file systems and file clients is not intended for general application use.

OO File Access: The File resource manager of the Open Blueprint provides OO access to file data through the Open Blueprint Persistence Services resource manager. Thus, OO applications have access to file, as well as database data.

It should also be noted that Persistence Services support for file data will coexist with procedural interfaces for file data. This accommodates customer evolution from existing procedural applications to object-oriented applications. The Open Blueprint Persistence Services support for record file data is a long-term objective.

Distributed File Access

Distributed file access with the Open Blueprint File resource manager is accommodated through the use and support of file clients and file exporters, as illustrated in Figure 4 on page 6. The structure illustrated provides the basis for supporting various file access protocols that are common in the industry.

DFS Distributed File Access

The Distributed File Services (DFS) protocols are designed for the distributed computing environment for scalable access to byte files in the enterprise. DFS employs the Open Blueprint RPC communications manager for communicating across the enterprise, and integrates well with other Open Blueprint services (for example, Directory, Identification and Authentication, Access Control and Common Transport Services).

The DFS capabilities include:

- Support for byte file access
- Support for a wide variety of file exporters
- Client caching with change notification
- Server-to-server caching with change notification
- File replication for maintaining multiple copies of data
- Support for file administration services

DFS servers may be configured to effect “local” caching of file data for performance. The DFS protocols and functions for cache notification are particularly well-suited to this purpose.

DFS servers may also be configured to effect “local” replication of file data for both performance or availability reasons. Replication of data allows local access to data and thereby reduces the concurrent access demand on any one server. Replication also increases the availability of the data through multiple copies of the data.

Note: File caching and replication can be important for multimedia support in large networks. Rather than having all users of the multimedia data concurrently accessing a single server, the demand can be effectively distributed to multiple DFS servers, each of which would support a smaller, limited number of users.

DDM Distributed File Access

The Distributed Data Management (DDM) protocols are designed for scalable, distributed access to record files in the enterprise. DDM employs the Open Blueprint Conversational communications manager for communicating across the enterprise, and integrates with other Open Blueprint services (for example, Identification and Authentication, Access Control and Common Transport Services).

Note: The integration of DDM support with Open Blueprint directory services is a long-term objective.

The DDM capabilities include:

- Support for record file access
- Client caching
- Server to server caching
- Support for file administration services

Other File Access Protocols

The Open Blueprint places emphasis on DFS and DDM file access protocols in that they emphasize integration with other Open Blueprint services and are pervasive across system platforms. However, effective participation in heterogeneous distributed environments requires support for other protocols as well.

The role of the support for other file access protocols is significant, and should not be overlooked. The Open Blueprint File resource manager structure accommodates support for multiple file protocols, but the protocols supported tend to vary by system platform and File resource manager implementation, based on customer requirements for the specific platforms.

In general, support for multiple file protocols is driven by the need to incorporate local or limited distributed configurations into an enterprise or cross-enterprise distributed configuration. Thus, the primary goal of support for protocols other than DFS and DDM is to support existing client environments with little or no modification to those client environments.

For example, providing file serving support to an existing UNIX configuration without modifying the existing UNIX clients would typically require that the file server provide file exporter support for the NFS file protocol. While a customer may choose to install DFS client support on the UNIX clients, the Open Blueprint distributed file support would not require it.

In this context, it is more important that the NFS support provided by the Open Blueprint File resource manager reflect what the NFS client expects to see (rather than File resource manager functions that go beyond what NFS protocols support). Thus, the goal for other protocol support, such as the support for NFS protocols is to accommodate NFS client access to data managed by Open Blueprint File resource managers, rather than prescribing changes that an NFS client (or NFS client applications) would have to make to gain access to Open Blueprint-managed files.

By the same token, the NFS client function in the Open Blueprint is designed to support those functions recognized by an NFS server.

Gateway File Servers

A variation of the structure for distributed file access that can be particularly useful is the concept of a gateway file server. In enterprise environments composed of multiple, distributed servers a file client may gain access to remote servers through a gateway server, rather than through direct connection to every server. An example of a gateway server is illustrated in Figure 8.

Gateway servers can be configured to bridge from one distributed file protocol to another. The most common use of this would be bridging “other file protocols” to a network of DFS servers. Rather than deploying multiple file exporters on all file servers, a gateway server can be configured to translate various “local” file protocols to DFS protocols. Figure 8 illustrates a gateway server that is configured to translate SMB and NFS request to DFS requests.

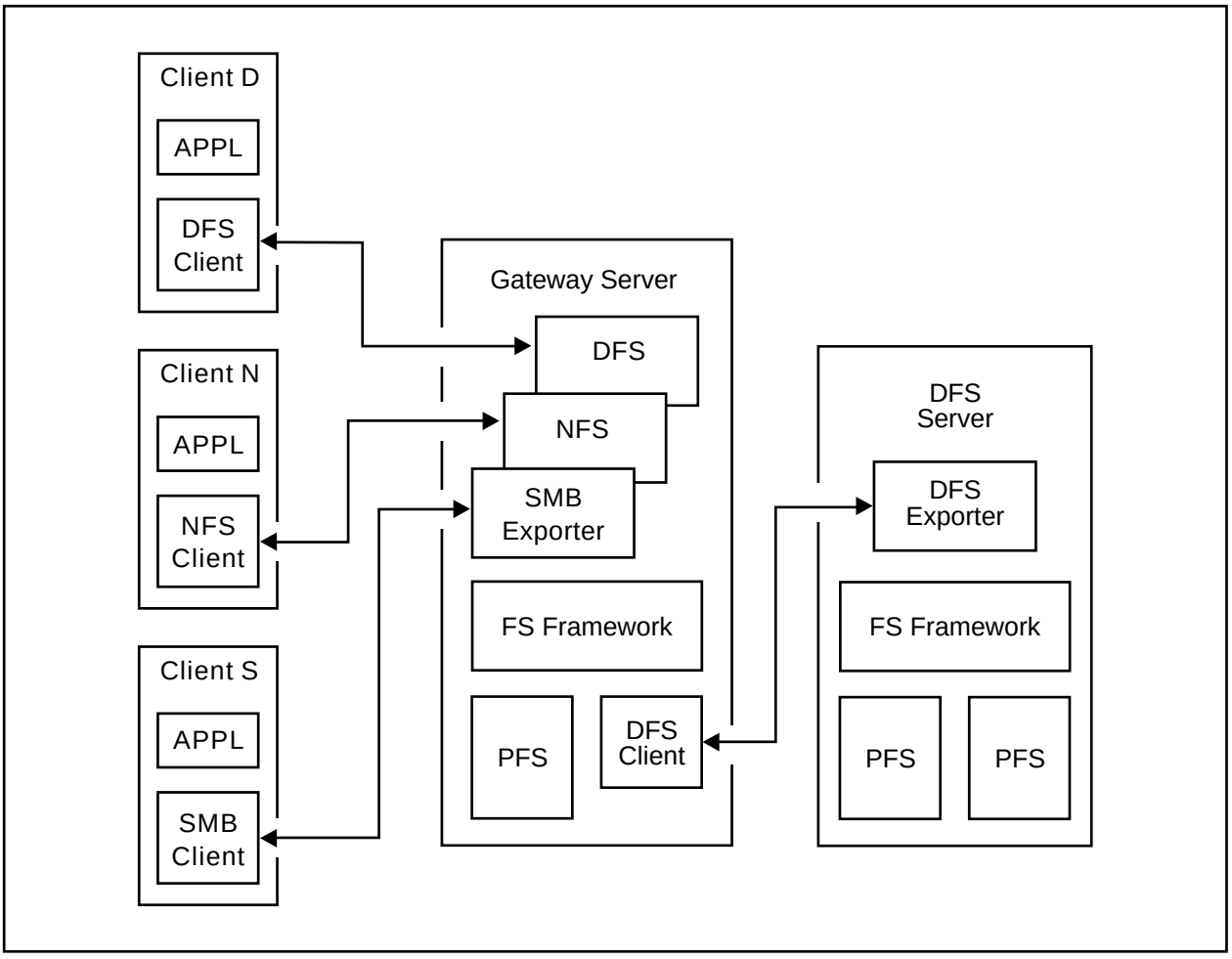


Figure 8. File Access Through Gateway Servers

Related Open Blueprint Resource Managers

The File resource manager utilizes the following components of IBM's Open Blueprint to effect integration with the overall distributed computing environment:

Directory

The File resource manager utilizes the Open Blueprint distributed directory services to register root directories as a distributed resource. The file system framework utilizes registered file systems to determine where to route file system requests.

Directory services hold information on the location of the file system. When used in a client/server configuration, the directory services allow the client to locate the server from the context of the path name to the file.

Note: File resource manager integration with the Open Blueprint directory services is key to supporting requirements for location and user independence (as identified in “File System Requirements and Background” on page 2).

Access Control

The File resource manager uses the Access Control resource manager of the Open Blueprint to allow controlled access to file resources.

Once authentication has been done, the file system must authorize and police access to its data. Access is policed by user credentials to a file for List, Read, Write, Update, Extend, Create, and Delete operations.

Identification and Authentication

The File resource manager integrates with the Open Blueprint Identification and Authentication services to exploit the single user sign-on context established by the client. Distributed file access does not require a separate sign-on or user enrollment.

In addition, the client and server must prove their identities to each other.

Note: File resource manager integration with the Open Blueprint Identification and Authentication services is key to supporting requirements for user location independence (as identified in “File System Requirements and Background” on page 2).

Communication Managers

File access protocols supported by File Exporters utilize the communication service applicable to the file protocol in question (for example, RPC for DFS, Conversational for DDM, and Transport Services for NFS). By utilizing the Open Blueprint Communications resource manager and the underlying use of Common Transport Semantics, it is possible to perform distributed file access over any common network transport.

Storage Management

The File resource manager relies on the Storage Management resource manager for administration and maintenance of storage and retrieval of inactive file data. Inactive storage services refer to maintaining backups and archives of data, and the corresponding services for recovering damaged data or retrieving archived data.

Transaction Monitor

For transaction environments, the File resource manager record file support integrates with the Open Blueprint Transaction Monitor services. On-line transaction monitors provide ACID[1] transaction properties to their applications. Thus, the Open Blueprint File resource manager record support must in its turn provide these to the Transaction Monitor.

---

[1] Atomicity, Consistency, Isolation, Durability properties of units of work as defined by the International Standards Organization (ISO).

Transaction Manager

For transaction environments, the integrated File resource manager record file and transaction monitor support utilizes the Open Blueprint Transaction Manager services for transaction coordination, logging and concurrency, as follows:

- Transaction processing

  The File resource manager participates in dynamic transaction processing as one of the coordinated resources. The File resource manager registers with the Transaction Manager and participates in two-phased commit processing.

- Concurrency Services

  Locking is done at the record level and a lock is held with the intention of performing some operation on a record or range of records. The locking capability of the file system is what allows each application using the file to be isolated from the effects of another application using the same file.

- Logging & Recovery Services
  - Updates to record file data are logged to support recovery processing.
  - Transaction state information is logged to support synchronization with the transaction coordinator on recovery processing.
  - Supports forward recovery, which is the ability to use a journal to roll forward all committed data from a previous backup.

File Administration Services

One of the key challenges to be addressed by the File resource manager is the management of file data in environments that support multiple, different file systems on multiple, distributed systems. The administration of file data can be broken down into two key elements:

1. Administration of Active storage
2. Administration of File Systems

The support for administration of file data in a distributed environment must address remote administration of multiple server systems, but retain the ability to control which administrators have control over what servers and data in the network.

Administration of Active Storage

Administration of active storage refers to the management of storage device resources that support file systems. In this area, the File resource manager supports two solutions:

- **Logical Volume Management** - The separation of device storage management for “data manager” or PFS implementation. A single volume manager is defined to support multiple data manager implementations (database as well as file systems).
- **Storage Hierarchy Services** - The ability to manage direct access storage for file systems on a hierarchy of storage devices and storage technologies. Storage hierarchy services are addressed by a separate paper (see *Open Blueprint Storage Management Resource Manager*).

Administration of logical volume storage is, in general, a function specific to the system (server) that holds the data server. However, the logical volume manager exports administration services to enable remote administration. The functions supported include:

- Partitioning of storage devices
- Defining and altering logical volumes
- Allocating storage to data managers (for example, file systems)
- Monitoring storage activity (that is, IO activity to storage constructs)
- Querying configuration information

File Systems

Administration of file systems is, in general, a function specific to the system (server) that runs the file server. However, the File resource manager exports administration services to enable remote administration. The functions supported include:

- Defining and altering file systems
- Monitoring file system activity
- Registering file systems in the distributed network
- Authorizing access to files
- Backup and restore of the file system without requiring the file system to be off-line while backup is taking place
- Import and export of the file system data, on a per file basis and in a format that can be used to move data between file systems
- Querying and updating the security characteristics of the file system

Notices

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

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

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

Trademarks

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

IBM
IBMLink
CICS
LANDP
Open Blueprint

The following terms are trademarks of other companies:

OSf	Open Software Foundation, Inc.
NFS	Sun Microsystems, Inc.
Encina	Transarc Corporation

Communicating Your Comments to IBM

If you especially like or dislike anything about this paper, please use one of the methods listed below to send your comments to IBM. Whichever method you choose, make sure you send your name, address, and telephone number if you would like a reply. Feel free to comment on specific error or omissions, accuracy, organization, subject matter, or completeness of this paper.

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.

- If you prefer to send comments by FAX, use this number:

  United States and Canada: 1-800-227-5088.

- If you prefer to send comments electronically, use one of these ID's:
  - Internet: **USIB2HPD@VNET.IBM.COM**
  - IBM Mail Exchange: **USIB2HPD at IBMMAIL**
  - IBMLink: **CIBMORCF at RALVM13**

Make sure to include the following in your note:

- Title of this paper
- Page number or topic to which your comment applies

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

GC23-3917-01