

**/DESCRIPTION OF A FILE ACCESS PROTOCOL
FOR COMPUTER NETWORKS/**

by

LARRY EDWARD PELLETIER

B. S., Loyola University of Los Angeles, 1970

-

A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

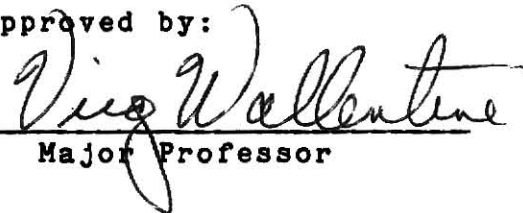
MASTER OF COMPUTER SCIENCE

Department of Computer Science

**KANSAS STATE UNIVERSITY
Manhattan, Kansas**

1985

Approved by:


Major Professor

LD
2668
R4
1985
P44
C. 2

TABLE OF CONTENTS

A11202 964544

1.0 Introduction	1
2.0 Concepts	7
2.1 Application Example	7
2.2 The Server	9
2.3 Users of the Server	10
2.4 Flow of Inter-machine Communication	12
2.5 Messages and Fields	13
2.6 Status Fields	14
2.7 Commands	14
2.8 Use of Attributes	15
2.9 Catalogs	16
2.10 Access Methods	16
2.11 Separation of Logical and Physical Descriptions	18
2.12 Open-ended Physical Formats	18
2.13 Application Walkthrough	19
3.0 Logical Formats	22
3.1 Terminology	22
3.2 Message Header	22
3.3 Logical Command Formats	25
3.3.1 File Commands	25
3.3.2 User Commands	26
3.3.3 Catalog Commands	27
3.3.4 Utility Commands	27

**THIS BOOK
CONTAINS
NUMEROUS PAGES
WITH DIAGRAMS
THAT ARE CROOKED
COMPARED TO THE
REST OF THE
INFORMATION ON
THE PAGE.**

**THIS IS AS
RECEIVED FROM
CUSTOMER.**

3.4	Command Summaries	29
3.4.1	File Commands	29
3.4.2	User Commands	33
3.4.3	Catalog Commands	34
3.4.4	Utility Commands	35
3.5	Common Field Descriptions	38
3.6	Attribute Descriptions	48
3.6.1	Catalog Attributes	48
3.6.2	Server Attributes	52
3.6.3	User Attributes	57
3.7	Status Values	57
3.7.1	Common Values	58
3.7.2	File Commands	58
3.7.3	User Commands	60
3.7.4	Catalog Commands	60
3.7.5	Utility Commands	61
4.0	Physical Formats	63
4.1	Field Implementation	63
4.2	Data Types	64
4.3	Physical Field Descriptions	65
4.4	Physical Attribute Descriptions	74
5.0	Summary	78
6.0	References	80

TABLE OF FIGURES

2-1 CALENDAR APPLICATION - APPLICATION FILES	7
2-2 CALENDAR NETWORK	8
2-3 PROTOCOL ENVIRONMENT	9
2-4 END TO END PROTOCOL DEFINITION	12
2-5 EXAMPLE PROGRAM FLOW	21

ACKNOWLEDGMENTS

I would like to thank Dr. Wallentine of Kansas State University for his guidance and input. I would also like to thank Ray Jantz, Raul Cartaya and Diana Foley of NCR for their input and review of the technical portion of this paper.

SNA and VSAM are trademarks of IBM

UNIX is a trademark of AT&T Bell Laboratories

CP/M is a trademark of Digital Research

ETHERNET is a trademark of Xerox

IMOS is a trademark of NCR

1.0 INTRODUCTION

The purpose of this paper is to define a File Access Protocol suitable for use with OSI (Open Systems Interconnection)[4], SNA (System Network Architecture)[14], ETHERNET[15] or some other network protocol. OSI is an International Standards Organization (ISO) standard for the exchange of information between different computer systems. OSI is organized into seven layers: the Physical Layer, which provides the physical connections between systems; the Data Link Layer, which provides control and error correction for the physical layer; the Network Layer, which provides routing services for the connections; the Transport Layer, which provides end to end control and optimized data transfer between system entities; the Session Layer, which organizes and synchronizes exchanges between system entities; the Presentation Layer, which normalizes the syntactic view of data transferred between systems; and the Application Layer, which contains the corresponding entities. The protocol described in this paper would reside in the Application Layer of OSI since it provides logical communication between a normalized file system and an application or end user.

SNA (System Network Architecture) is IBM's network description for communicating between multiple IBM (and IBM compatible) systems. It is organized into seven layers: the Link Layer, which corresponds to OSI's physical layer; the Data Link Control Layer, which corresponds to OSI's data link layer; the Path Control Layer, which corresponds to OSI's network and transport layers; the Transmission Control and Data Flow Control Layers which correspond to OSI's session layer; the Presentation Services Layer which corresponds to OSI's presentation layer; and the End User Layer, which corresponds to OSI's application layer. This protocol would reside in the End User Layer of SNA.

Ethernet is a Local Area Network specification[15] supported by Xerox, DEC, and Intel. It is generally described as a layered architecture along the lines of OSI but only the layers corresponding to OSI's physical and data link layers have been standardized. The other layers are vendor dependent. This protocol would reside in these upper layers.

This protocol provides complete file management facilities for a user on the network. The protocol supports record oriented functions, file oriented functions, catalog oriented functions, utility functions and user oriented functions. The protocol supports these functions for disc, tape and printer operations. Cross device operations are

also supported. Access rights and concurrency control are included within the protocol. The functions described in this paper were compiled through a survey of the following sources: CP/M(Control Program/Monitor)[12], an operating system for micro computers from Digital Resource Corp.; UNIX[9,10,11], an operating system for micro and mini computers from AT&T(Bell Labs); IMOS[13] an operating system for mini computers from NCR; VSAM(Virtual Storage Access Method)[1,2], a file management system for mainframe computers from IBM; ANSI(American National Standards Institute) COBOL[3], a programming language with complete file management facilities; and ISO's File Transfer, Access and Management Application Layer protocol for OSI[5,6,7,8].

The supported file functions include: select and de-select, which cause a file to be chosen and released; open and close, which ready a file for processing and shut it down; and copy and concatenate, which cause files to be duplicated and combined. The record functions include: read, which causes a record to be transferred from a file to memory; write, which causes a new record to be transferred from memory to a file; rewrite, which causes a record to be replaced in a file; position, which moves the point of reference within a file; and lock and unlock, which provide concurrency protection. The catalog functions include configure catalog and query catalog, which allow the user to

change file catalogs and retrieve information about files. The utility functions include: server attach and server detach, which establish and release a session with the server; back-up, archive and restore, which allow files to be saved and restored from magnetic tape; de-spool, which allows stored print files to be printed; media-initialize, which allows fresh removable media such as magnetic tape or floppy discs to be readied for processing; and server-configure, and server-query, which allow the user to change and inquire about the characteristics of the server. The user functions include: sign-on and sign-off, which identify the user to the server; and user-configure and user-query; which allow the user to change and inquire about his own characteristics as the server sees them.

A list of attributes describe the server and each user and catalog associated with the server. An example of a server attribute is the server's free space, which indicates how much storage is still available on the server. An example of a user attribute is the user password, which provides a security check when the user signs on. An example of a catalog attribute is the creation time, which identifies the date and time the catalog was created. These attributes are accessible for change or viewing via the configure and query commands for the server, user and catalog.

This paper contains four sections. This introduction section is followed by a section on concepts, a section on logical formats and a section containing an example physical format.

The concepts section contains a discussion of the various concepts around which this paper, the protocol and the functionalism are built and an example of how it would be used in an application. Some of the concepts that are covered are: the server, users of the server, flow of inter-machine communication, status, commands, messages and fields, use of attributes, catalogs, access methods, concurrency control, access rights, division of protocol description into logical and physical parts, and open ended physical formats.

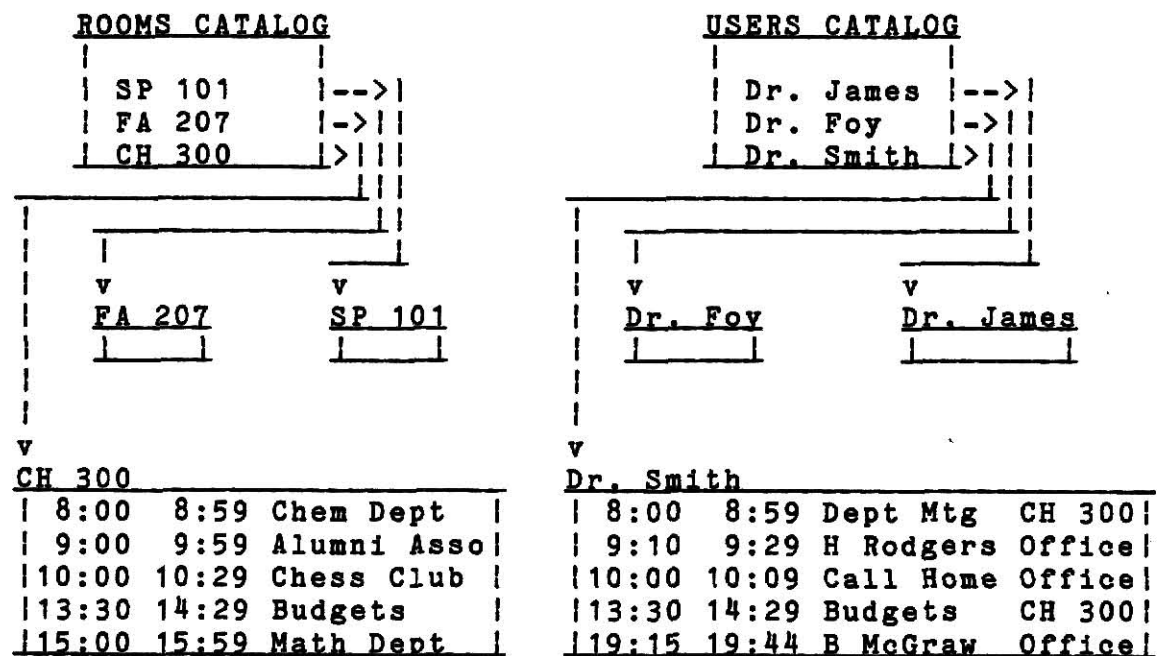
The logical formats section describes all logical parts of the protocol. This section includes a description of message headers, the logical formats of the various commands, a summary of the functionalism of each command, and a description of the parameters used by the commands. It also includes a description of the attributes that may be associated with servers, catalogs and users, and a list of possible return status.

The physical formats section describes a possible physical implementation for a typical network. It includes a description of the physical representation of messages and fields and lists of the physical values assigned to fields and attributes.

2.0 CONCEPTS

2.1 APPLICATION EXAMPLE

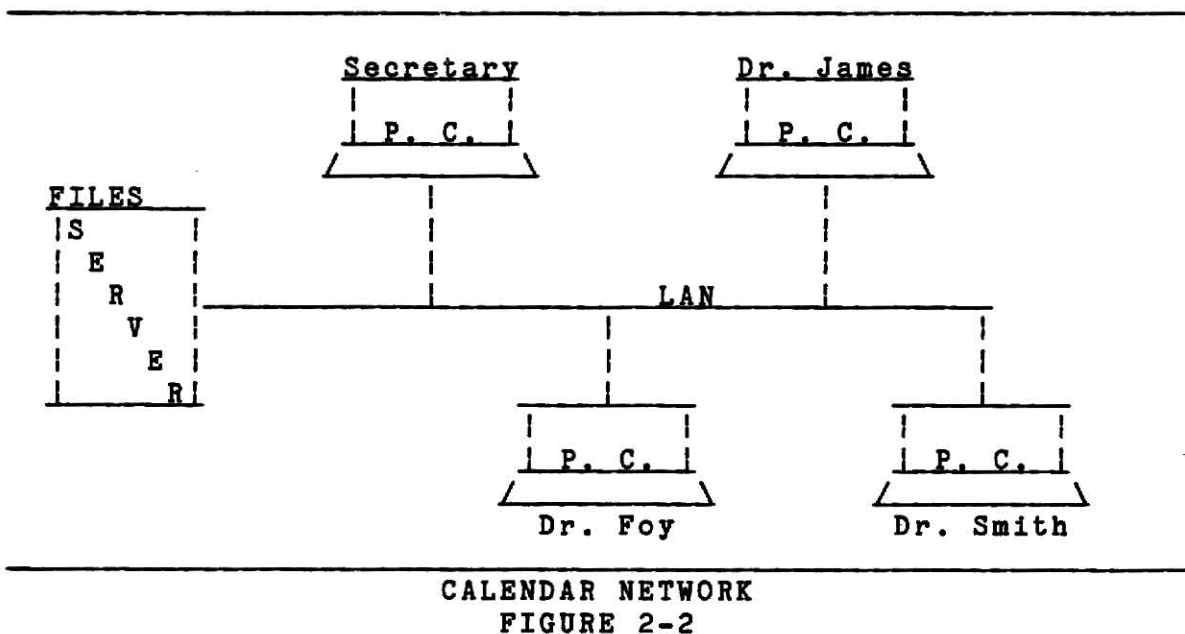
Throughout this section a computerized calendar application will be used as an example for showing how the protocol concepts may be applied. The application consists of a set of programs and files that interact with a set of users allowing them to set up meetings, appointments and reminders, and display or print schedules on a daily or weekly basis.



CALENDAR APPLICATION - APPLICATION FILES
FIGURE 2-1

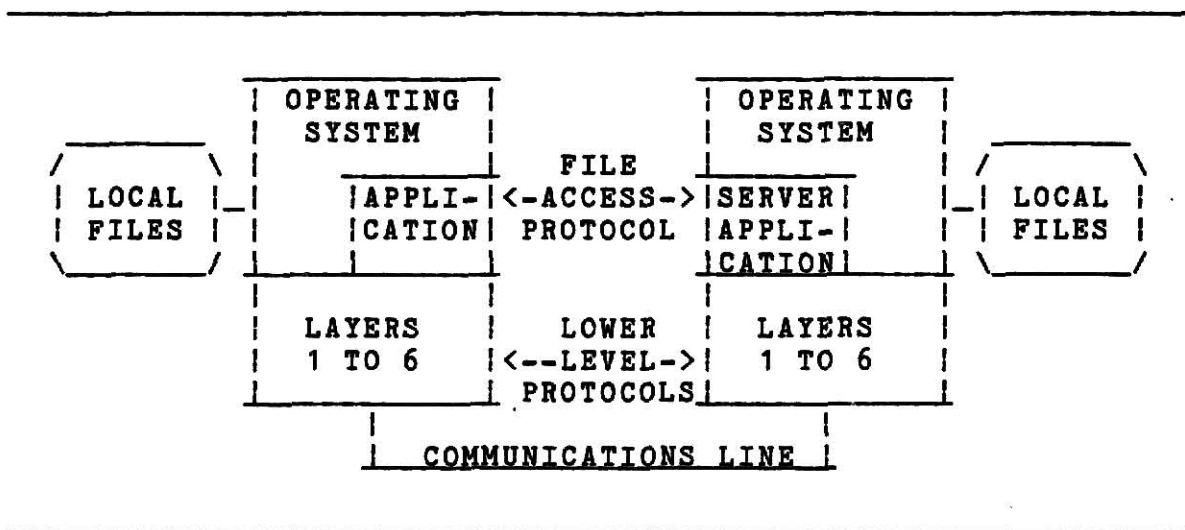
A central catalog contains the user's names. Another contains a list of meeting rooms. Each user has a corresponding file that contains his schedule. Each room has a corresponding file associated with it. Both these files are indexed files. The records contain a description of a time slot with the record key being the date and starting time of a scheduled event.

For the purpose of this paper, it will be assumed that this calendar application is implemented on a network with the files at one location (called the server) and the programs run from many locations (such as personal computers in the users' offices).



2.2 THE SERVER

The protocol defined by this paper is for a multi-computer network such as our example in figure 2-2. When combined with appropriate lower level protocols such as those defined by ETHERNET[15] or OSI[4] this protocol will allow entities associated with computers on the network such as the calendar programs to control and access files in other computers on the network. See figure 2-3.



PROTOCOL ENVIRONMENT
FIGURE 2-3

Whether these entities use this protocol directly or indirectly through translation of the computer's normal file management functions is beyond the scope of this paper. However, some mechanism will exist on that computer that will drive the protocol.

The computer that controls the files contains an entity that handles requests. This entity is referred to as the server. Whether the server is a special application or a utility for handling these requests or an integrated part of the operating system, is again beyond the scope of this paper.

This protocol, then, defines the conversation between requesting entities and the servers across the network. It can also be used to normalize the differences between file management systems of different computers.

2.3 USERS OF THE SERVER

The users of the server are divided into three classes: Systems, Requestors and Users. These are directly associated with information the server uses to fulfill requests made upon it. The information is location, concurrency control and access control.

The System (i.e. host, application processor or intelligent terminal) is the location on the network that requests to the server are coming from. It may provide requests for a single requestor/end-user combination, or for multiple requestors and multiple end-users, or for everything in between. In our calendar example, a system is one of the personal computers in the professors' offices.

It provides requests for the program running in the personal computer and the person running the program.

The Requestor signifies the unit of concurrency for the host system. Each requestor from a system will signify a concurrent thread for that system. Requestors may be associated with a job, program, process, or task. There may even be only one requestor per host system on some uni-processing systems. In our example, the Requestor is the program running in the personal computer and accessing the calendar files.

The User is the entity that has access rights to objects in the server. Objects include files, catalogs, and server functions. Access rights include read, write, modify, delete and execute. The user has an optional password associated with it and may have other passwords signifying permission to use an access right/object combination. Within the system the user may be a terminal operator, a program submitter, a program or a system process. In the calendar example, the user is the professor who is currently running the calendar application. No matter what office and what personal computer he or she runs on, that professor is always considered the same user and has the same access rights.

2.4 FLOW OF INTER-MACHINE COMMUNICATION

The requestors and servers communicate with each other as peers. In order to perform a file management function, the communications between them must be orderly and reliable. This requires that this protocol must be defined as an end to end protocol [14,16].

(Request-Data-to-Follow)	----->	
Request-Complete	----->	
	<-----	(Intermediate-Response)
(Inquiry-Request)	----->	
(Suspend-Request)	----->	
	<-----	(Intermediate-Response- w/Suspension)
(Continue-Request)	----->	
(Cancel-Request)	----->	
	<-----	Final-Response

END TO END PROTOCOL DEFINITION
FIGURE 2-4

An end to end protocol between the requestor and the server is a series of requests and replies. The requests and replies are implemented as a series of messages sent from the requestor to the server and vice versa. Each request with response or responses and any later request-

follow up for the request is identified by a unique number for that requestor on the system. Figure 2-4 shows the possible request messages and response messages that can be contained in a series. Optional messages are in parentheses.

Any number of "Request-Data-to-Follow" messages may precede the "Request-Complete" message. The other optional messages may occur at any time between the "Request-Complete" and the "Final-Response" except the "Continue-Request" can only occur after an "Intermediate-Response-w/Suspension".

2.5 MESSAGES AND FIELDS

A message is the implementation of a request or reply. It contains the information that is passed back and forth between requestor and server. The request from the requestor may be for reading a record in a file. The reply from the server would be the data in the record. The message consists of a group of fields, some of which make up the envelope of the message while other fields make up the body of the message. The envelope fields contain control information that is implemented in lower level protocols in layered protocols.

Each field contains a logically different part of the

information in a message. Fields may be implemented as an offset into the message or via a separator or via an identifier. Some of the types of information that may be contained in fields are: message identifiers, command types, command parameters, status and response data. Fields are also used in this paper to implement attributes. For field definitions see the message header, common field descriptions, and attribute descriptions sections of this paper.

2.6 STATUS FIELDS

Status fields are part of every response message. Multiple status are allowable. There are three types of status: Intermediate, Suspension, and Final. Only intermediate status are allowed in an intermediate response. Intermediate status may be included in an intermediate response with suspension and a final response. At least one suspension status must be included in an intermediate response with suspension and at least one final status must be included in a final response.

2.7 COMMANDS

A command tells the server what the requestor wishes the server to do. An example of a command is a request to read a record of a file. The command set described in this paper was chosen to provide functionality with a variety of

existing file systems. These range from primitive file systems such as provided by the personal computer operation system CP/M[12] to sophisticated file systems such as VSAM[11]. A special emphasis is placed on the requirements for the standard COBOL language[3] as it contains a sophisticated file access description for a widely used language. Other commands were selected to provide support for a distributed processing environment such as is described by OSI[4] (Open System Interconnection). Commands to address the problems of individual files distributed physically across the network and coordinating the updates of these distributed files are not included as these problems are beyond the scope of this paper.

2.8 USE OF ATTRIBUTES

While parameters of commands allow the manipulation of object characteristics, there is a need for some of these characteristics to span the scope of individual commands. This need is fulfilled by the attribute concept. Attributes are characteristics of objects that are more permanent than the scope of one command such as the creation date of a file. The objects that have attributes defined in this paper are: servers, users, catalogs and files. Attributes are global, permanent fields within the scope of the server. They are set by or during one command and used by or during

another. Some attributes are only set by the server implicitly. They are read only attributes. Other attributes may be modified under certain conditions. A special set of commands is provided to view attributes and to explicitly add, change and delete them. These are the query and configure commands.

2.9 CATALOGS

Catalogs contain lists of files. This allows the requestor to group logically related files together. A catalog is implemented as a file itself as required by UNIX[9]. This allows catalogs to be organized in hierarchies. A current catalog is kept for each user of the server. Each server has a unique catalog that is the root of the hierarchy for that server.

2.10 ACCESS METHODS

An access method is a logical approach used for reading and writing files. The protocol defined by this paper supports four access methods.

The simplest access method is the sequential access method. It is required by the most basic operating systems such as CP/M[12] and by file transfer protocols such as the one defined by OSI[5]. In the sequential access method, access starts with the first record of the file and proceeds

in sequential order to the last record of the file. The server keeps track of the current record.

A variation of the sequential access method is the FIFO (first in, first out) access method. This method is used by UNIX[10] to implement its standard I/O. In the FIFO access method, reads are always from the beginning of the file and include a delete of the record. Writes are always to the end of the file. Files accessed via the FIFO access method may be used as queues and may be used effectively to coordinate processes in different systems that have access to the server.

The random access method is a more sophisticated access method. It is also required by basic operating systems such as CP/M[12]. In the random access method, records may be accessed in any order. Records are accessed by record number or by a record key (which may be indexed).

The dynamic access method is a combination of the sequential and random access methods. It is required by ANSI COBOL[3] and by sophisticated operating systems such as IMOS[13] and VSAM[1]. In the dynamic access method, records may be accessed in any order by key as the random access method or a record may be located by key via a position command and then records may be accessed sequentially as in the sequential access method.

2.11 SEPARATION OF LOGICAL AND PHYSICAL DESCRIPTIONS

In this paper, the description of the logical aspects of the protocol are separated from the physical description of the protocol. The logical description of the messages, commands, parameters, attributes etc. without committing to how they are implemented in terms of bits and bytes. The physical description takes the logical description and shows how it is to be implemented in bits and bytes. This separation of logical and physical description allows one logically distinct protocol to be implemented in different physical environments in a manner efficient to the environment. If these physically different implementations are required to communicate with each other, translation can be achieved in a presentation layer such as OSI layer six[4]. This translation is eased by the fact that the implementations are based on the same logical definition.

2.12 OPEN-ENDED PHYSICAL FORMATS

An open-ended physical format provides for the future inclusion of new commands, new fields and new attributes. One example of an open-ended physical format is a parsable free format as will be used in this paper. This allows smooth migration from release to release and the sharing of a common interface among hosts of different capabilities.

2.13 APPLICATION WALKTHROUGH

In order to illustrate how this protocol works, we will use a possible program from the calendar application. This program would allow a user to add a reminder to his calendar. An example flow for the program is shown in figure 2-5. The highlighted steps require access to the server. It is assumed that the system (personal computer) that the program is running on is already attached to the server.

In the first highlighted step(1), the users id (name) and access password is passed to the server in a user sign on command as a request-complete message. The good status is returned (assuming the user and password are valid) in a final response message. In this step a concurrency thread for the program as a requestor is established in the server. In all following steps the command will be passed in a request-complete message and the status returned in a final response message.

In (2), the catalog is readied for use as an indexed file via an open command. If the catalog is available a good status will be returned. In (3), catalog is searched for the user's file via a read command with the user's name as the index key. If the record is found, a good status is returned along with the record data in a final response message. An intermediate response of record unavailable at

this time might be returned first if another requestor is using the record. In (4), the catalog is released via a close command.

In (5), the user file is opened as an index file much like the catalog was opened in (2). In (6), a read command is sent to the server. In this read, the index key is the time and date but the search is made for a match or the record with the next earliest time. The start and stop times in this record can be compared with the event time to see if there is a conflict. If there is no earlier event a beginning of file intermediate status would be sent along with the record not found final status. Both would be sent in a final response message. In (7), the command to write a record might be sent in a request-data to follow message, with the record data sent in a request-complete message. In (8), the server is told that the user file is no longer needed for processing.

In (9), the server is told via a user sign off command that the user will no longer access the server via this program. This program's concurrency thread on the server is then relinquished.

```

      START PROGRAM
      v
      GET USER'S NAME
      v
      GET USER'S PASSWORD
      v
      SIGN USER ON TO SERVER(1)
      v
      OPEN CATALOG OF USERS(2)
      v
      READ USER'S ENTRY IN CATALOG(3)
      v
      DOES ENTRY EXIST? -> NO ----->|
      v YES
      CLOSE CATALOG OF USERS(4)
      v
      OPEN USER FILE(5)
      SAY INVALID USER
      ----->v<-----
      ^
      GET DATE FOR REMINDER
      v
      GET TIME FOR REMINDER
      v
      GET EVENT TO BE REMINDED OF
      v
      READ USERS FILE FOR EVENT
      AT OR PROCEEDING REMINDER TIME(6)
      v
      IS THERE AN EXISTING EVENT? -> NO---->|
      v YES
      NOTIFY USER OF CONFLICT
      WRITE REMINDER
      YES
      INTO FILE(7)
      ^<- ASK IF EVENT TO BE RESCHEDULED?
      | NO
      v<-----
      ASK IF MORE REMINDERS TO BE SCHEDULED? -> YES --->|
      v NO
      CLOSE USER FILE(8)
      v<-----
      SIGN USER OFF OF SERVER(9)
      v
      END OF PROGRAM

```

EXAMPLE PROGRAM FLOW
FIGURE 2-5

3.0 LOGICAL FORMATS

3.1 TERMINOLOGY

In describing the logical formats the following conventions are used:

[] signifies an optional field

(|) signifies an either/or situation

{ } signifies one or more instances may occur

The commands are expressed in the following format:

Command (Input fields)(Output fields)

3.2 MESSAGE HEADER

All messages, no matter what low level protocols they are built on, must contain certain information. This information may be physically in the lower level protocols or in the message header itself. For the purpose of this paper, the required information will be specified as the message header.

Logical Format

Host-id Requestor-id Message-Number Message-Flag
[Command-Type] [Message-Data]

Field Descriptions

Host-id: A unique identifier of the processor that sent the message.

Requestor-id: A unique identifier per host-id of the requestor that sent the message. The granularity of the requestor (host, end-user, program, process, task) is host dependent.

Message-Number: A unique identifier per requestor to identify a unique request and its response.

Message-Flag: Message flag identifies the direction and purpose of the message. It may assume the following values:

Request-Complete: This value signifies a complete request message or the last of a series of messages making up a request.

Request-Data-to-follow: This value signifies a request message that is not the last message.

Inquiry-Request: The inquiry requests the status of a previous request. It uses the same message number as that request.

Cancel-Request: The cancel requests processing to stop on a previous request. It uses the same message number as that request.

Suspend-Request: The suspend requests processing to stop on a previous request until a Continue request is sent.

Continue-Request: The continue requests processing to continue on a message that was suspended by the server or the requestor.

Intermediate-Response: This response returns a status and implies that the service requested is continuing.

Intermediate-Response-w/suspension: This response returns a status and implies that the service requested will remain suspended until the requestor sends a continue.

Final-Response: This response returns a status and implies that the service requested is finished.

Command-Type: The command type is valid only on the initial request message. The command section will contain the different commands.

Message-data: The message data contains all other fields. This includes command parameters, status and response data.

3.3 LOGICAL COMMAND FORMAT

3.3.1 FILE COMMANDS

Select ([User-Instance] File-Name [Select-Type] [File-Type] [File-Password] [Allocation] [Organization] [Rec-Type] [Exp-date] [Max-Rec-Size] [[[Key-id] Key-Position Key-Length]] [Reference-Rights] [Select-Rights]) (Status File-Instance)

Print-Select [User-Instance] (Queue | Printer-Type) [Media-Name]) (Status File-Instance)

De-Select (File-Instance [Delete-Flag]) (Status)

De-Select-All () (Status)

Open (File-Instance [Select-Type] [File-Type] [File-Password] [Organization] [Rec-Type] [Exp-Date] [Max-Rec-Size] [[[Key-id] Key-Position Key-Length]] [Reference-Rights] [Select-Rights] [Open-Type] [Share-Model] [Access-Model]) (Status)

Close (File-Instance) (Status)

Position (File-Instance ([Key-id] Record-Key [Match-Key-Condition] | Record-Number) [Direction]) (Status [Record-Key | Record-Number])

Read (File-Instance [[Key-id] Record-Key | Record-Number] [No-Records | Blocking-Size]) (Status {Record-Length Record-Content} [Record-Key | Record-Number])

Write (File-Instance [[Key-id] Record-Key | Record-
 Number] [No-Records] {Record-Length Record-
 Content}) (Status [Record-Key | Record-Number])
 Rewrite (File-Instance [[Key-id] Record-Key | Record-
 Number] Record-Length Record-Content) (Status)
 Delete (File-Instance [[Key-id] Record-Key | Record-
 Number] [No-Records]) (Status [Record-Key |
 Record-Number])
 Lock (File-Instance [[Key-id] Record-Key | Record-
 Number] Lock-Type) (Status)
 Unlock (File-Instance [[Key-id] Record-Key | Record-
 Number]) (Status)
 Unlock-All (File-Instance) (Status)

3.3.2 USER COMMANDS

User-Sign-on (User-id [User-Password]) (Status User-
 Instance)
 User-Sign-off (User-Instance) (Status)
 User-Configure (User-Instance {Action Object Value})
 (Status)
 User-Query (User-Instance {Object} [Output-File-Name [File-
 Password]]) (Status [{Object Value}])

3.3.3 CATALOG COMMANDS

Catalog-Configure ([User-Instance] Catalog-Name [Catalog-Password] File-Name [File-Password] {Action (Object Value)}) (Status)

Catalog-Query ([User-Instance] Catalog-Name [Catalog-Password] {File-Name [File-Password] {Object}} [Output-File-Name [File-Password]]) (Status [{File-Name {Object Value}}])

3.3.4 UTILITY COMMANDS

Copy ([User-Instance] Destination-File-Name [File-Password] Source-File-Name [File-Password] [Different]) (Status)

Back-Up-File ([User-Instance] {File-Name [File-Password]} (Tape-Name | Tape-id)) (Status)

Restore-File ([User-Instance] {File-Name [File-Password]} (Tape-Name | Tape-id)) (Status)

Despool-File ([User-Instance] {File-Name [File-Password] [Media-Name]} (Queue | Printer-Type)) (Status)

Media-Initialize ([User-Instance] Device-id Media-Format Media-Name [Media-Text]) (Status)

Server-Configure ([User-Instance] {Action Object Value}) (Status)

Server-Query ([User-Instance] {Object} [Output-File-Name [File-Password]]) (Status [{Object Value}])

Acrhive ([User-Instance] File-Name [File-Password] [Tape-
Name | Tape-id) (Status)
Attach ([System-Password]) (Status)
Detach () (Status)

3.4 COMMAND SUMMARIES

3.4.1 FILE COMMANDS

Select This command fulfills the requirement of OSI file protocols[6] and operating system JCLs[2,13] of creating a relationship between the user and a file. Selecting a file readies it for subsequent open and record access. Access rights checking takes place at this time. The file is checked and protected with regards to simultaneous access at this time. Creation parameters may be supplied at this time or at "open" time. A unique "file instance" is returned to identify this usage of the file by this "user". This identifier must be used for subsequent calls for this instance.

Print-Select

This command is a special case of select that provides printer support. A print select creates a printer type file which is automatically routed to the specified print "queue" or "printer type" which must be assigned to a queue. Optionally a "media name" may be specified. This is the name of the special form to be mounted while this file prints. A unique "file instance" is returned to identify this usage of the file by this "user".

This identifier must be used for subsequent calls for this instance.

De-Select This command fulfills the requirement of OSI file protocols[6] and operating system JCLs[2,13] of removing a relationship between the user and a file. De-selecting destroys a "file instance" and all concurrency controls associated with it. The physical file may be deleted at this time, if security allows, via use of the "delete flag".

De-Select-All

All selected files for a requestor may be de-assigned via this command. Deletion is not allowed.

Open This command fulfills a requirement common to operating systems[1,11,12,13], network file protocols[6] and ANSI COBOL[3]. Opening a particular "file instance" readies it for record access. The access mode, I/O type, and share mode if applicable are specified at this time.

Close This command fulfills a requirement common to operating systems[1,11,12,13], network file protocols[6] and ANSI COBOL[3]. When a particular "file instance" is closed, that instance is no longer available for record access. All record resources for the instance are released.

Position This command provides the functionality provided by the COBOL[3] start, VSAM[1] point and UNIX[11] seek commands. A position is used to position a dynamically accessed "file instance" prior to sequential access and specify direction of that access. The "file instance" may be positioned at, before, or after a particular record depending on file type.

Read This command fulfills a requirement common to operating systems[1,11,12,13], network file protocols[6] and ANSI COBOL[3]. A read operation on a file returns the content of a particular record and optionally the contents of a specified number of records following that record. In the dynamic or sequential mode, if no particular record is specified the record content will be for the next or last sequential record(s) depending on specified direction.

Write This command fulfills a requirement common to operating systems[1,11,12,13], network file protocols[6] and ANSI COBOL[3]. A write operation on a file places the specified record content into a previously non-existing record or set of records. In the dynamic or sequential mode for non-sequential files, if no particular record is specified,

the record content will be placed after the last or next existing record found sequentially from the current position depending on direction.

Rewrite This command fulfills a requirement common to operating systems[1,11,12,13], and ANSI COBOL[3]. A rewrite operation on a file places the specified record content into a previously existing record. If no particular record is specified the record content will be placed in the current record.

Delete This command fulfills a requirement common to operating systems[1,11,13], and ANSI COBOL[3]. A delete operation on a file makes the specified record(s) non-existent. If no particular record is specified, the current record(s) is made non-existent. Records in a file opened as FIFO may not be deleted.

Lock This command provides user controlled concurrency control. A specified record may be locked using a "lock-type" of either exclusive or shared. If no particular record is specified, the current record is locked.

Unlock This command provides user controlled concurrency control. The lock on the specified record is released. If no particular record is specified the current record is unlocked.

Unlock-All

This command provides user controlled concurrency control. All records for the requestor on the specified file are unlocked. If no file is specified, all record locks on all files for this requestor are released.

3.4.2 USER COMMANDS

User-Sign-On

This command provides for multiple users accessing the server simultaneously. The sign on function allows the end user to identify himself to the server for security checks. After the user is signed on, his "user instance" can be used with other commands to clear security checks.

User-Sign-Off

This command provides for multiple users accessing the server simultaneously. After a user signs off, his "user instance" is no longer valid in security checks.

User-Configure

This command provides for control of multiple users. Defaults and Options for a user may be changed via a configure command. An "action" (e.g. add, delete, change) is performed on one of

the user's "objects" (e.g. password) given a "value".

User-Query

This command provides for control of multiple users. The "value"(s) of specified user's "object"(s) may be obtained via query. The "value"(s) may optionally be placed in a specified file.

3.4.3 CATALOG COMMANDS

Cat-Configure

This command supports the catalog concept supported by UNIX[11] and the virtual filestore concept of OSI file protocols[6]. Information about the files contained in a catalog may be added, changed or deleted via a configure command. This includes adding and deleting files themselves. An "action" (e.g. add) is performed on one of the catalog "objects" (e.g. exp-date) given a "value". Some operations can be performed only in specific circumstances. See Catalog Attributes for specifics.

Catalog-Query

This command supports the catalog concept supported by UNIX[11] and the virtual filestore concept of OSI file protocols[6]. Information about the

files contained in a catalog may be retrieved via a query command. The "value"(s) of specified catalog "object"(s) may be obtained via query. The "value"(s) may optionally be placed in a specified file.

3.4.4 UTILITY COMMANDS

Copy This command fulfills a requirement common to operating systems[2,10,12,13]. The copy creates a copy of the source file in the destination file. If "different" is specified the destination will be placed on a different physical unit, if possible. "Don't care" characters are valid in the file name(s).

Back-Up-File

This command fulfills a requirement common to operating systems[2,10,12,13]. The specified file is copied to the specified tape. If a catalog file is specified, all files in that catalog will be backed up. "Don't care" characters are valid in the file name(s).

Restore-File

This command fulfills a requirement common to operating systems[2,10,12,13]. The specified file is restored from the specified tape. If a catalog

was backed up, all files will be restored. "Don't care" characters are valid in the file name(s).

Despool-File

This command fulfills a requirement common to operating systems[10]. The specified file is routed to the printer queue or printer type specified. Optionally a media name, signifying the form to be used while printing, may be specified. Only files with a "file type" of "Printer" may be selected. "don't care" characters are valid in the file name(s).

Media-Initialization

This command fulfills a requirement common to operating systems[10,13]. The removable media on the specified device is initialized in the specified "media format" with the specified "media name" and the optional "media text".

Server-Configure

This command supports the concept of server attributes. The server's configuration, defaults and options may be modified. An "action" is performed on one of the system "objects" with a given "value". The "object" may optionally be protected with a password.

Server-Query

This command supports the concept of server attributes. The "value"(s) of specified system "object"(s) may be obtained via query. The "value"(s) may be placed in a specified file.

Archive This command fulfills a requirement common to operating systems[10,12]. All files in the given catalog that have an archive attribute and have been changed since the last archive of that catalog will be backed up. If no tape is specified the tape specified as the catalog's attribute will be used. "file name" must be a catalog file.

Attach This command provides the connect functionality of OSI file protocols[8]. The first access from a system must be by an attach.

Detach This command provides the dis-connect functionality of OSI file protocols[8]. After a system detaches, no more requests from that system will be recognized.

3.5 COMMON FIELD DESCRIPTIONS

Access-Mode This field specifies the kind of access which a requesting program will use. Valid values are:

Sequential - Records to be accessed in order

Random - Records to be accessed by a key or record number

Dynamic - Records to be accessed either in order or by a key or record number (default)

FIFO - Records to be read and deleted from the beginning of the file and written to the end of the file

Action This field describes the kind of operation to be performed in a configure command. Valid values are:

Add Delete Change

Allocation This field designates a file's original allocation size in bytes. It may be set in a new file at or before the first open of the file. If this field is not specified, no space will be allocated to the file until the first write.

Blocking-Size This field contains the maximum number of bytes that may be used to contain logically sequential complete records starting with the one specified in the request.

Catalog-Name This field is a special case of "File-Name" that applies to a catalog. See "File-Name".

Catalog-Password

This field contains the password that allows the current user to access the specified catalog.

Delete-Flag The presence of this flag in the de-select command indicates that the file should be deleted. The flag is ignored if the file is selected by other users.

Destination-File-Name

This field describes a file to be used as a destination in an operation that requires source and destination files. See "File-Name".

Device-id This field contains a logical identifier for a physical device.

Different The presence of this field specifies that the destination file be placed on a different physical unit from the source.

Direction	<p>This field specifies whether the file is to be accessed in the forward or backward direction for subsequent sequential access. Valid values are:</p> <div> <div>Forward</div> <div>Backwards</div> </div>
Exp-Date	<p>This field contains the date on which this file becomes invalid and automatically deleted. Absence of a date means automatic deletion will not occur.</p>
File-Instance	<p>This field is a unique identifier of a particular instance of a file. It is used to identify what file instance a particular operation is to be performed on.</p>
File-Name	<p>This field contains the full path file name from the current catalog or one of its ancestors to the file. The catalog names and the file name are all separated from each other by the name separator. (See HXX-Name-Separator in the server attributes section.)</p>
File-Password	<p>This field contains the password that allows the current user to access the specified file.</p>
File-Type	<p>This field is used to define file types that require special handling. Valid values are:</p> <div> <div>Normal(default)</div> <div>Spool</div> <div>Catalog</div> </div>

Key-id This field contains a unique identifier for each key in a multi-key index file. It is used only in multi-key index files.

Key-Length This field is used with indexed files. It specifies the length of the key in bytes.

Key-Position This field defines the position (relative to zero) within a data record where the key begins.

Lock-Type This field indicates whether a lock is shared or exclusive. Items locked as shared may have multiple owners. Items locked as exclusive may have only one owner. Valid values are:

Shared

Exclusive

Match-Key-Condition

This field is used to position a data file or catalog. The file is searched until the following condition is true: "Primary Key" "Match-Key-Condition" "Match Key" where "Primary Key" is the portion of the record that identifies it and "Match Key" is the record key, record number or file name. The valid values are:

Equal

Greater-Than

Not-Less-Than

Max-Record-Size

This field specifies, in bytes, the actual record length for fixed length records and the maximum record length for variable length records.

Media-Format This field specifies what format the removable media at the specified location should be initialized at.

Media-Name This field specifies the identifier for the removable media of the special print form.

Media-Text This field specifies what text should be placed in the media header. The exact contents depend on what format the media was initialized with.

No-Records This field indicates the number of additional logically sequential records that are to be included in the request.

Object This field describes what an "action" in a configure command will be performed upon. The "objects" are attributes of the system and users. Valid "objects" are described in the user attributes, catalog attributes and server attributes sections.

Open-Type This field describes the kind of capability requested for an opened file. Valid values are:

Input - File to be opened for input only (default - new files)

Output - File to be opened for output only

I/O - File to be opened for input and output (default - old files)

Extend - Records to be added to the end of the file

Organization This field describes a file's internal organization, as viewed by the requestor. Valid values are:

Sequential Relative (default) Indexed

Output-File-Name

This field describes a file in which the information from the server is to be contained. See "File-Name".

Printer-Type This field describes a user defined, generic printer type to be used to pick a printer for file despooling. Each "Printer-Type" is associated with a queue. "Printer-Type"s are assigned via "Server-Configure" command.

Queue	This field specifies a server print queue.
Rec-Type	<p>This field indicates whether a file's data records are fixed or variable in length.</p> <p>Valid values are:</p> <p style="margin-left: 40px;">Fixed-Length-Records (default)</p> <p style="margin-left: 40px;">Variable-Length-Records</p>
Record-Content	This field contains the contents of a record.
Record-Key	<p>This field is used in record I/O messages to identify the current or desired position within a file. It contains the key value of a record in an indexed or catalog file. For random I/O requests, it represents the key of the desired record. For sequential read responses, it represents the key of the record read. For position responses and certain failed responses, it specifies the current position of the file as seen by the server.</p>
Record-Length	<p>This field contains the length of the "Record-Content" field.</p>
Record-Number	<p>This field is used in record I/O messages to identify the current or desired position within a file. It contains the relative</p>

record number for files with relative organization. For usage see description of "Record-Key".

Reference-Rights

This field specifies the degree of file interlocking for this file instance. Valid values are:

Exclusive - only one accessor at a time. (default)

Shared - multiple accessors reading and writing the file.

Read-only - multiple accessors only reading the file.

Select-Rights This field specifies the selection access rights accorded to other users of a file by the owner of the file. Valid values are:

Sharable - This file can be selected by other users. (default)

Private - This file can be selected only by the owner.

Select-Type This field specifies the manner in which a file is to be used. Valid values are:

New - The file is to be created. It must not already exist.

Old - The file must already exist.
(default)

Replace - The file is to be created.
If it already exists it will
be deleted first.

Guarantee - If the file does not exist,
it will be created.

Share-Mode This field specifies what kind of locking is
to be performed. Valid values are:

Explicit - Lock requests come from
requestor (default)

Implicit - Locks supplied by server

Source-File-Name

This field describes a file to be used as a
source in an operation that requires source
and destination files. See "File-Name".

Status This field contains an intermediate or final
status in a response message. For valid
values see separate status section.

System-Password

This field contains the password that allows
the requesting system to sign on to the
server.

Tape-id This field describes a "Device-id" that
applies to a tape. See "Device-id".

Tape-Name	This field contains the name by which a tape may be identified. For tapes it is synonymous with "Media-Name". See "Media-Name".
User-id	This field contains the information that uniquely identifies an end user. The end user may be a person, process, program, job or system that has a set of attributes and capabilities.
User-Instance	This field uniquely identifies an end user/requestor combination that has cleared security checks at sign on.
User-Password	This field contains the password that allows the specified user to access the server.
Value	This field contains the contents of system or user attributes as required by the configure and query commands. For a list of attributes and their values see the catalog attributes, user attributes and server attributes sections.

3.6 ATTRIBUTE DESCRIPTIONS

3.6.1 CATALOG ATTRIBUTES

Name	This field contains the name of the file or catalog described by this record. This name is the name known within the parent catalog and does not include ancestor catalog names.
File-Type	See "File-Type" in command field descriptions. This attribute may be modified only if size is equal to zero.
Allocation	See "Allocation" in command field descriptions. This attribute may be modified only if size is less than allocation.
Organization	See "Organization" in command field descriptions. This attribute may be modified only if size is equal to zero.
Rec-Type	See "Rec-Type" in command field descriptions. This attribute may be modified only if size is equal to zero.
Exp-Date	See "Exp-Date" in command field descriptions. The format is YYMMDD where YY is year, MM is month and DD is day.
Max-Rec-Size	See "Max-Rec-Size" in command field descriptions. This attribute may be modified only if size is equal to zero.

Key-id	See "Key-id" in command field descriptions. This attribute may be modified only if size is equal to zero. This field may have more than one appearance.
Key-Position	See "Key-Position" in command field descriptions. This field occurs once for every key id. This attribute may be modified only if size is equal to zero.
Key-Length	See "Key-Length" in command field descriptions. This field occurs once for every key id. This attribute may be modified only if size is equal to zero.
Size	This field contains the size of the file in bytes including index space and fill space for inactive records. This is a read only attribute.
Data-Size	This field contains the size of the active data for this file in bytes. This is a read only attribute.
Index-Size	This field contains the size of all indexes for this file in bytes. This is a read only attribute.
Creation-Time	This field contains the date and time that the file was created. This is a read only attribute. The format is YYMMDDhhmmss where

YY is year, MM is month, DD is day, hh is hour, mm is minutes, ss is seconds.

Last-Use-Time This field contains the date and time that the file was last accessed. This is a read only attribute. For format see "Creation-Time".

Last-Mod-Time This field contains the date and time that the file was last modified. This is a read only attribute. For format see "Creation-Time".

Last-Archive-Time

This field contains the date and time that the file was last archived. This is a read only attribute. For format see "Creation-Time".

No-Opens This field contains a count of the number of times the file has been opened since creation. This is a read only attribute.

Archive-Tape This field contains the tape name of this default tape for archiving a catalog. This is valid only for catalog entries.

Archive This field indicates whether the file is to be archived. Valid values are:

True False

No-Records This field contains a count of the number of active records the file contains. This is a read only attribute.

Deleted-Records

This field contains a count of the number of deleted records the file contains. This is a read only attribute.

Owner This field contains the user id of the user who owns this file. The creator in the original owner. This attribute may be read or modified only by the owner.

User-id This field contains the user id of a user who is allowed permission on this file. This attribute may be read, added or modified only by the owner. This field may have more than one occurrence.

Read This field indicates whether the associated user may read the file. This attribute may be added or modified only by the owner. It occurs once for every user of the file and may only be modified or added or read when the associated user id is modified, added or read. Valid values are:

True False

Write	This field indicates whether the associated user may write to the file. Valid values, and usage are the same as for "Read".
Execute	This field indicates whether the associated user may execute the file. Valid values, and usage are the same as for "Read".
Append	This field indicates whether the associated user may append additional records to the file. Valid values, and usage are the same as for "Read".
Password	This field contains the password that allows the associated user to access the file according to the associated read, write, append and execute permissions. It occurs once for every user id and may be modified, added or read when the associated user id is modified, added or read. Itr may also be modified by the user.

3.6.2 SERVER ATTRIBUTES

Server-Identity

This field provides a unique identity for the server.

No-Files	This field contains a count of the number of files currently being managed by this server. This is a read only attribute.
No-Open-Files	This field contains a count of the number of files managed by this server that are currently open. This is a read only attribute.
Total-Space	This field contains the total capacity available from this server excluding removable media. This is a read only attribute.
Free-Space	This field contains a count of the number of bytes of free space available from this server excluding removable media. This is a read only attribute.
Password	This field contains the password that allows systems to attach to the server.
Device-id	This field contains a unique identifier for each peripheral on the server. This field may occur multiple times.

DXX-Type This field contains the type of device that device "XX" is. Valid values are:

- Sequential-Output-Device (Printer)
- Sequential-Device (Tape)
- Fixed-Random-Device (Disk)
- Removable-Random-Device (disk)

DXX-State This field describes the current state of device "XX". This is a read only field. Valid values are:

- Ready
- Not-Ready
- Empty (Removable media only)
- Unknown

DXX-Media-Name This field is valid for removable media and printers only. It contains the name of the media or form currently on device "XX".

DXX-No-I/O's This field contains a count of the number of I/O's to device "XX" since the last time this statistic was gathered. This is a read only attribute.

DXX-Free-Space This field contains the number of bytes of unallocated space on device "XX". This is a read only attribute. It may not apply to some device types.

DXX-Max-Size This field contains the size in bytes of the maximum amount of data that may be stored on this device. This field may not apply to some device types.

QXX-Printer-Id This field contains the device id of a printer on which queue XX's files may be printed. More than one device id may be specified per queue. If no device id is specified, the files may be printed on any ASCII printer.

QXX-Printer-Type

This field contains a printer type that is assigned to queue XX. More than one printer type may be specified.

QXX-Priority This field contains a priority specifying how important it is that queue XX's files be printed. The lower the priority number the more important the queue.

QXX-Flag This flag indicates whether queue XX is active or on hold. Valid values are:

Active Hold

Host-id This field contains a unique identifier for each host that may request services from the server. This field may occur multiple times.

HXX-State This field describes the current state of host "XX". This is a read only field. Valid values are:

Ready Not-Ready Unknown

HXX-No-Requests

This field contains a count of the number of requests that have been received from host "XX" since the last time this statistic was gathered. This is a read only attribute.

HXX-DC-Character

This field specifies what character is a "Don't Care" character replacement for Host "XX". The default value of this field is "?".

HXX-DC-String This field specifies what character is a "Don't Care" string replacement for Host "XX". The default value of this field is "\$".

HXX-Name-Separator

This field specifies what character is to be used to separate catalog and file names for Host "XX". The default value of this field is "/".

3.6.3 USER ATTRIBUTES

User-id	This field contains an identifier that is unique for each end user of this server. The user id is independent of host id or requestor id.
No-Requests	This field contains a count of the number of server requests that this user has issued since the last time this statistic was gathered. This is a read only attribute.
Home-Catalog	This field contains the catalog currently being used as default by this user.
Password	This field contains the password that allows this user to sign on to the server.

3.7 STATUS VALUES

In the following description of status values, the common values section contains those statuses that may be applicable to all commands. The other sections list statuses applicable to that family of commands with valid individual commands listed for each status. Following each status is an indication as to whether the status is an intermediate, suspension or final status (e.i. I, S, F).

3.7.1 COMMON VALUES

Operation-Initiated	(I)
Operation-Proceeding-	(I)
Normally	
Unknown-Parameter(s)	(I)
Operation-Suspended	(S)
Operation-Successful	(F)
Server-Busy	(F)
Command-Unknown	(F)
Insufficient-Parameters	(F)
Parameter-Value-Illegal	(F)
Access-Violation	(F)
Password-Violation	(F)
Server-Internal-Error	(F)
No-Storage-Available	(F)
Operation-Cancelled	(F)
Operation-Partially-	(F)
Completed	

3.7.2 FILE COMMANDS

File-Created	(I)	Select, Open
File-Selected	(I)	Select, Open
File-Unavailable-at-	(I)	Select, Open
this-Time		
File-Replaced	(I)	Select, Open
Record-Truncated	(I)	Write, Rewrite

Beginning-of-File	(I)	Position, Read, Write
End-of-File	(I)	Position, Read, Write, Delete
Record-Unavailable-at- this-Time	(I)	Read, Write, Rewrite, Delete, Lock
File-Closed	(I)	Deselect, Deselect-All, Close
File-Deleted	(I)	Deselect
File-Not-Deleted	(F)	Deselect
User-Instance-Unknown	(F)	Select, Open
File-Not-Found	(F)	Select, Open
File-Already-Selected	(F)	Select
File-Already-Exists	(F)	Select, Open
File-Parameters-Mismatch	(F)	Select, Open
File-Already-Opened	(F)	Open
File-Not-Extendable	(F)	Open
File-Instance-Unknown	(F)	All except Select, Deselect-All
File-in-Unrecovered-State	(F)	All
Record-Not-Found	(F)	Position, Read, Write, Rewrite, Delete, Lock, Unlock
File-Boundary-Error	(F)	Position, Read, Write, Rewrite, Delete, Lock

File-Not-Opened	(F)	Position, Read, Write, Rewrite, Delete, Lock, Unlock, Unlock-All
Operation-Cancelled- Deadlock	(F)	Read, Write, Rewrite, Delete, Lock
Primary-Key-Duplicated	(F)	Write

3.7.3 USER COMMANDS

File-Created	(I)	User-Query
File-Closed	(I)	User-Sign-Off
User-id-Unknown	(I)	User-Sign-on
Object-Unknown	(F)	User-Configure, User-Query
User-Instance-Unknown	(F)	User-Configure, User-Query, User-Sign-Off

3.7.4 CATALOG COMMANDS

Catalog-Unavailable-at- this-Time	(I)	All
File-Created	(I)	Catalog-Query
User-Instance-Unknown	(F)	All
Catalog-Not-Found	(F)	All
Object-Unknown	(F)	All
File-Not-Found	(F)	All

3.7.5 UTILITY COMMANDS

File-Created	(I) Copy, Restore-File, Server-Query
File-Unavailable-at- this-Time	(I) Copy, Back-Up-File, Despool-File, Archive
No-Different-Device- Available	(I) Copy
Device-Unavailable-at- this-Time	(I) Back-Up-File, Archive Restore-File, Despool- File, Media-Initialize
Media-Unavailable-at- this-Time	(I) Back-Up-File, Restore-File
Tape-at-End-of-Reel	(S) Back-Up-File, Archive, Restore-File
Printer-Out-of-Paper	(S) Despool-File
Device-Inoperative	(S) Back-Up-File, Archive, Restore-File, Despool- File, Media-Initialize
User-Instance-Unknown	(F) All except Attach, Detach
File-Not-Found	(F) Copy, Back-Up-File, Despool-File, Archive
Object-Unknown	(F) Server-Query, Server-Configure

File-in-Unrecovered-State	(F)	Copy, Back-Up-File Despool-File, Archive
Device-Unknown	(F)	Back-Up-File, Archive Restore-File, Despool- File, Media-Initialize
Media-Unknown	(F)	Back-Up-File, Archive, Restore-File
Format-Unknown	(F)	Despool-File, Media-Initialize
Unknown-Print-Queue	(F)	Despool-File

4.0 PHYSICAL FORMATS

4.1 FIELD IMPLEMENTATION

All fields and attributes are described by a field header that identifies the field or attribute and specifies its length. Field headers precede the field.

Special fields are used to group together related fields and to separate groups of repeating fields. These are the "start group" and "end group" fields. They may be used recursively. Group fields must be used whenever {} appears in the logical message. They may also be required in some instances that are described below in the physical field definitions.

The format of the field header is as follows:

byte 1: bits 0-7: 8 bits of field ID.

byte 2: bit 0: 0 - Length is 7 bits

1 - Length is 15 bits

bits 1-7: High order 7 bits (or all) of
field length.

byte 3: bits 0-7: Low order 8 bits of field length.

NOTE: byte 3 is omitted if bit 0
of byte 2 is off.

4.2 DATA TYPES

This section describes the data types specified for fields and attributes in the following sections.

Alphanumeric

This is an ASCII string that allows alphabetic, numeric and special characters. If less than maximum is specified, filling is done with spaces on the right and truncation is from the right.

Binary This field contains an unsigned binary number. If less than maximum is specified, filling is done with binary zeros on the left and truncation is from the left.

Boolean This is always a one byte field. Odd binary numbers mean "true" (least significant bit on). Even binary numbers mean "false" (least significant bit off).

Data This field contains unformatted data.

Date This field has the format YYMMDDhhmmss, where:

YY is years ranging from ASCII 0 to 99

MM is months ranging from ASCII 1 to 12

DD is days ranging from ASCII 1 to 31

hh is hours ranging from ASCII 0 to 23

mm is minutes ranging from ASCII 0 to 59

ss is seconds ranging from ASCII 0 to 59

Hexadecimal

This field contains a unsigned hexadecimal number.
If less than maximum is specified, filling is done
with hex 00 on the left and truncation is from the
left.

4.3 PHYSICAL FIELD DESCRIPTIONS

This section provides the physical characteristics of
the fields described in the logical formats section. These
descriptions are in the format:

Field Name	Code	Data type	Length range
------------	------	-----------	--------------

If logical field values were listed in the logical
format section, they will be listed here with their physical
values.

Field Descriptions:

Acess-mode	59	alphanumeric	1 byte
Sequential		S	
Random		R	
Dynamic		D	
FIFO		F	
Action	10	alphanumeric	1 byte
Add		A	
Delete		D	
Change		C	
Allocation	43	binary	1-4 bytes

Blocking-Size	68	binary	1-2 bytes
Catalog-Name	1C	alphanumeric	1-17 bytes
Command-Type	05	hexadecimal	1 byte
Archive		60	
Attach		01	
Back-Up-File		61	
Catalog-Configure		42	
Catalog-Query		43	
Close		34	
Copy		63	
Delete		3A	
De-Select		31	
De-Select-All		32	
Despool-File		65	
Detach		02	
Lock		3B	
Media-Initialize		64	
Open		33	
Position		35	
Read		37	
Restore-File		62	
Rewrite		39	
Select / Print-Select		30	
Server-Configure		12	
Server-Query		13	

Unlock	3C		
Unlock-All	3D		
User-Configure	72		
User-Query	73		
User-Sign-Off	71		
User-Sign-On	70		
Write	38		
Catalog-Password	Same as File-Password		
	(Must be used with brackets)		
Delete-Flag	5F	no value	0 bytes
Destination-File-Name	2F	alphanumeric	1-17 bytes
Device-id	20	binary	1 byte
Different	22	no value	0 bytes
Direction	69	alphanumeric	1 byte
Backwards		B	
Forwards		F	
End-Group	0E	no value	0 bytes
Exp-Date	44	date	2-6 bytes
File-Instance	0B	binary	1-2 bytes
File-Name	40	alphanumeric	1-17 bytes
File-Password	41	alphanumeric	1-8 bytes
File-Type	42	alphanumeric	1 byte
Normal		N	
Spool		S	
Catalog		C	

Host-id	01	hexadecimal	1 byte
Key-id	4C	alphanumeric	1-6 bytes
Key-Length	4D	binary	1 byte
Key-Position	4E	binary	1-2 bytes
Lock-Type	6F	alphanumeric	1 byte
Shared		S	
Exclusive		E	
Match-Key-Condition	6C	alphanumeric	1 byte
Equal		E	
Greater-Then		G	
Not-Less-Then		N	
Max-Record-Size	49	binary	1-2 bytes
Media-Format	25	alphanumeric	1-6 bytes
Media-Name	24	alphanumeric	1-8 bytes
Media-Text	26	alphanumeric	1-40 bytes
Message-Flag	04	hexadecimal	1 byte
Request-Complete		00	
Request-Data-to-Follow		10	
Inquiry-Request		20	
Cancel-Request		80	
Suspend-Request		C0	
Continue-Request		40	
Intermediate-Response		21	
Intermediate-Response-w/Suspension		31	
Final-Response		01	

Message-Number	03	binary	1-4 bytes
Object	11	hexadecimal	1-2 bytes
See attribute descriptions			
Open-Type	58	alphanumeric	1 byte
Input		I	
Output		O	
I/O		U	
Extend		E	
Organization	45	alphanumeric	1 byte
Sequential		S	
Relative		R	
Indexed		I	
Output-File-Name	1F	alphanumeric	1-17 bytes
Printer-Type	28	alphanumeric	1-6 bytes
Queue	29	alphanumeric	1 byte
Rec-Type	48	alphanumeric	1 byte
Fixed-Length-Records		F	
Variable-Length-Records		V	
Record-Contents	60	data	1-2048 bytes
Record-Key	61	alphanumeric	1-2048 bytes
Record-Length	Contained in length of Record-Content		
Record-Number	62	binary	1-4 bytes

Reference-Rights	52	alphanumeric	1 byte
Exclusive		E	
Shared		S	
Read-Only		R	
Requestor-id	02	hexadecimal	1-3 bytes
Select-Rights	50	alphanumeric	1 byte
Sharable		S	
Private		P	
Select-Type	51	alphanumeric	1 byte
New		N	
Old		O	
Replace		R	
Gaurantee		G	
Share-Mode	5A	alphanumeric	1 byte
Explicit		E	
Implicit		I	
Start-Group	0F	no value	0 bytes
Status	08	hexadecimal	1 byte
Access-Violation		0B	
Begining-of-File		9A	
Catalog-Not-Found		20	
Catalog-Unavailable-at-this-Time		A8	
Command-Unknown		5A	
Device-Inoperative		F0	
Device-Unavailable-at-this-Time		B9	

Device-Unknown	30
End-of-File	9B
File-Already-Exists	12
File-Already-Opened	1A
File-Already-Selected	19
File-Boundary-Error	1F
File-Closed	93
File-Deleted	94
File-Instance-Unknown	10
File-in-Unrecovered-State	16
File-Not-Deleted	18
File-Not-Extendable	14
File-Not-Found	11
File-Not-Opened	1B
File-Parameters-Mismatch	13
File-Replaced	92
File-Selected	91
File-Unavailable-at-this-Time	98
Format-Unknown	F6
Insufficient-Parameters	09
Media-Unavailable-at-this-Time	BA
Media-Unknown	34
No-Different-Device-Available	B8
No-Storage-Available	04
Object-Unknown	22

Operation-Cancelled	03
Operation-Cancelled-Deadlock	1C
Operation-Initiated	80
Operation-Partially-Completed	02
Operation-Proceeding-Normally	81
Operation-Successfull	00
Operation-Supended	C0
Paramater-Value-Illegal	0A
Password-Violation	0C
Primary-Key-Duplicated	1D
Printer-Out-of-Paper	F8
Record-Not-Found	17
Record-Truncated	99
Record-Unavailable-at-This-Time	9C
Server-Busy	01
Server-Internal-Error	07
Tape-at-End-of-Reel	FC
Unknown-Parameter(s)	88
Unknown-Print-Queue	38
User-id-Unknown	A0
User-Instance-Unknown	21

Tape-id	Same as Device-id		
Tape-Name	Same as Media-Name		
User-id	18	alphanumeric	1-8 bytes
User-Instance	0A	alphanumeric	1-4 bytes
User-Password	19	alphanumeric	1-8 bytes
Value	12		

Length, type and values depend on object

4.4 PHYSICAL ATTRIBUTE DESCRIPTIONS

This section provides the physical characteristics of the attributes described in the Catalog, Server and User Attributes sections as seen by the protocol. This does not necessarily reflect the physical formats of the attributes internal to the server. The attributes are "Objects" of the Configure and Query commands. These descriptions are in the format:

Attribute Name	Code	Data type	Length range
----------------	------	-----------	--------------

Attributes that include a host or device id have two byte codes. The second byte is the host or device id. It is represented by "xx" in the description.

If logical attribute values were listed in the logical format section, they will be listed here with their physical values.

Attribute Descriptions:

Allocation	43	binary	1-4 bytes
Append	87	boolean	1 byte
Archive	82	boolean	1 byte
Archive-Tape	83	alphanumeric	1-8 bytes
Creation-Time	8C	date	12 bytes
Data-Size	91	binary	1-4 bytes
Deleted-Records	99	binary	1-4 bytes
Device-id	20	binary	1 byte
DXX-Free-Space	A4xx	binary	1-4 bytes

DXX-Max-Size	A5xx	binary	1-4 bytes
DXX-Media-Name	ACxx	alphanumeric	5 bytes
DXX-No-I/O's	A8xx	binary	1-3 bytes
DXX-State	A2xx	alphanumeric	1 byte
Ready		R	
Not-Ready		N	
Empty		E	
Unknown		U	
DXX-Type	A0xx	hexadecimal	1 byte
Sequential-Output-Device		02	
Sequential-Device		01	
Fixed-Random-Device		80	
Removable-Random-Device		00	
Execute	86	boolean	1 byte
Exp-Date	44	date	6 bytes
File-Type	42	alphanumeric	1 byte
Normal		N	
Spool		S	
Catalog		C	
Free-Space	95	binary	1-4 bytes
Home-Catalog	88	alphanumeric	1-17 bytes
Host-id	01	hexadecimal	1 byte
HXX-DC-Character	B4xx	alphanumeric	1 byte
HXX-DC-String	B5xx	alphanumeric	1 byte
HXX-Name-Separator	B6xx	alphanumeric	1 byte

HXX-No-Requests	B8xx	binary	1-3 bytes
HXX-State	B0xx	alphanumeric	1 byte
Ready		R	
Not-Ready		N	
Unknown		U	
Index-Size	92	binary	1-4 bytes
Key-id	4C	alphanumeric	1-6 bytes
Key-Length	4D	binary	1 byte
Key-Position	4E	binary	1-2 bytes
Last-Archive-Time	8F	date	12 bytes
Last-Mod-Time	8E	date	12 bytes
Last-Use-Time	8D	date	12 bytes
Max-Record-Size	49	binary	1-2 bytes
Name	40	alphanumeric	1-17 bytes
No-Files	9C	binary	1-2 bytes
No-Open-Files	9B	binary	1-2 bytes
No-Opens	9A	binary	1-2 bytes
No-Records	6A	binary	1-4 bytes
No-Requests	98	binary	1-3 bytes
Organization	45	alphanumeric	1 byte
Sequential		S	
Relative		R	
Indexed		I	
Owner	80	alphanumeric	1-8 bytes
Password	41	alphanumeric	1-8 bytes

QXX-Flag	C2xx	boolean	1 byte
Active		True	
Hold		False	
QXX-Priority	C3xx	binary	1 byte
QXX-Printer-id	C0xx	binary	1 byte
QXX-Printer-Type	C1xx	alphanumeric	1-6 bytes
Read	84	boolean	1 byte
Rec-Type	48	alphanumeric	1 byte
Fixed-Length-Records		F	
Variable-Length-Records		V	
Server-Identity	8A	hexadeximal	1 byte
Size	90	binary	4 bytes
Total-Space	94	binary	1-4 bytes
User-id	18	alphanumeric	8 bytes
Write	85	boolean	1 byte

5.0 SUMMARY

This paper describes a file protocol that could be used with a variety of layered architectures. An example of how the protocol can be used with a distributed calendar application is included. This example shows how some of the more common functions of the protocol would be used in a program by describing the program flow.

The protocol provides a comprehensive file access functionality that can be used with various operating systems and in various environments. By interfacing different operating systems and environments to a common file access protocol, heterogeneous systems can share data. This will allow a large variety of equipment to be used together. This gives a user more flexibility in equipment selection.

The logical definition of the protocol is separated from the physical definition in this paper. This is to allow the same protocol to be implemented on different physical networks. Again this approach provides for a larger selection of equipment on which the protocol can be implemented. This also gives a user more flexibility.

This paper gives only an example of a physical implementation of the protocol. In future work, a physical implementation of the protocol could be defined for a particular network such as ETHERNET. The functions in this

paper were distilled by examining a limited number of source systems. These systems were selected to cover a very wide range of requirements; however, not all possible functions are included. In future work, the protocol could be expanded through inclusion of functions of other systems.

6.0 REFERENCES

1. OS/VS Virtual Storage Access Method (VSAM) Programmer's Guide, GC26-3838-3, 1978, IBM Corporation, San Jose, California.
2. OS/VS2 Access Method Services, GC26-3841-3, 1980, IBM Corporation, San Jose, California.
3. American National Standard programming language COBOL, ANSI X3.23-1974, American National Standards Institute Inc., New York, New York.
4. Information processing systems - Open systems interconnection - Basic reference model, DRAFT INTERNATIONAL STANDARD ISO/DIS 7498, 1982, International Organization for Standardization.
5. Working Draft on File Transfer, Access and Management - June 1982 - General Description, ISO/TC97/SC16 N1190, International Organization for Standardization.
6. Working Draft on File Transfer, Access and Management - June 1982 - The Virtual Filestore, ISO/TC97/SC16 N1222, International Organization for Standardization.
7. Working Draft on File Transfer, Access and Management - June 1982 - The File Service Definition, ISO/TC97/SC16 N1223, International Organization for Standardization.

8. Working Draft on File Transfer, Access and Management - June 1982 - The File Protocol Specification, ISO/TC97/SC16 N1224, International Organization for Standardization.
9. Rebecca Thomas PhD, Jean Yates, A USER GUIDE TO THE UNIX SYSTEM, 1982, OSBORNE/McGraw-Hill, Berkeley, California.
10. Richard Gauthier, USING THE UNIX SYSTEM, 1981, Reston Publishing Company Inc., Reston, Virginia.
11. UNIX TIME-SHARING SYSTEM: UNIX PROGRAMMER'S MANUAL, Seventh Edition, Volume 1, 1979, Bell Telephone Laboratories Incorporated, Murray Hill, New Jersey.
12. CP/M OPERATING SYSTEM MANUAL, 1982, Digital Research, Pacific Grove, California.
13. NCR IMOS COBOL Student Text, 1977, NCR Corporation, Dayton, Ohio.
14. Paul E. Green, COMPUTER NETWORK ARCHITECTURES AND PROTOCOLS, 1982, Plenum Press, New York.
15. The Ethernet - A Local Area Network, Data Link Layer and Physical Layer Specifications, 1980, INTEL CORPORATION, Santa Clara, California.
16. Davis, Barber, Price, Solomandies, COMPUTER NETWORKS AND THEIR PROTOCOLS, 1979, John Wiley and Sons, New York.

**DESCRIPTION OF A FILE ACCESS PROTOCOL
FOR COMPUTER NETWORKS**

by

LARRY EDWARD PELLETIER

B. S., Loyola University of Los Angeles, 1970

-

AN ABSTRACT OF A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF COMPUTER SCIENCE

Department of Computer Science

**KANSAS STATE UNIVERSITY
Manhattan, Kansas**

1985

ABSTRACT

This paper describes a high level file access protocol suitable for use with a variety of layered network architectures. A wide variety of systems was used to select the functionality for the protocol. An example of how the protocol may be used is given using a distributed calendar application. The concepts of servers, attributes, catalogs, access methods, concurrency control and access rights are discussed as they pertain to the defined protocol. The logical description of the protocol is separated from the physical description for ease of application in a variety of environments. An example physical description is included.