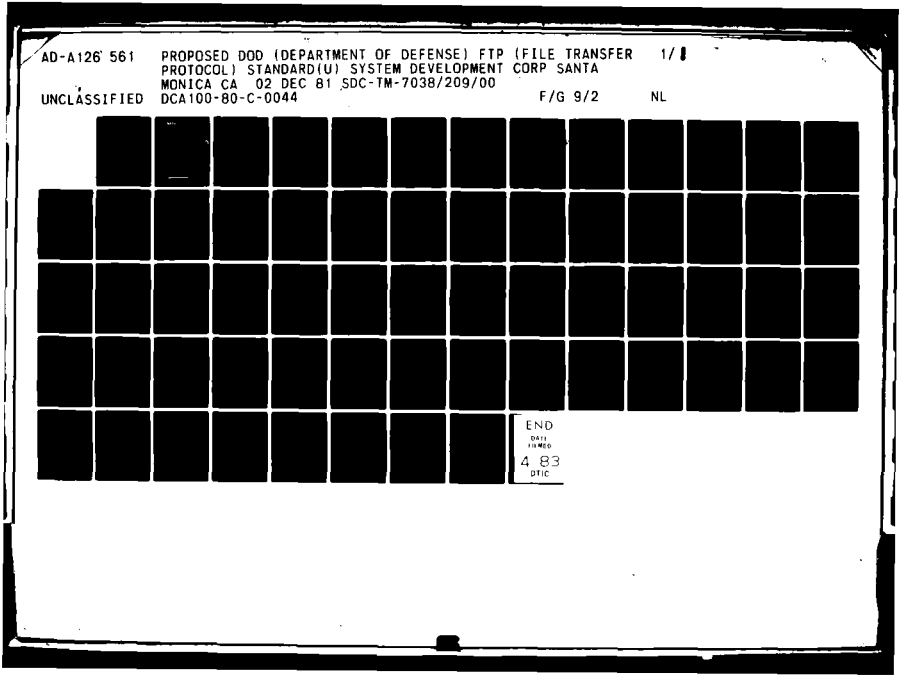
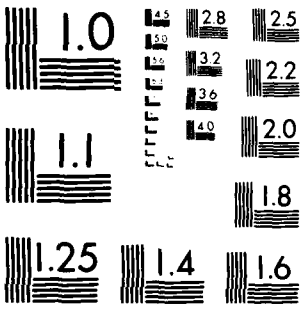

AD-A126 561
PROPOSED DOD (DEPARTMENT OF DEFENSE) FTP (FILE TRANSFER
PROTOCOL) STANDARD(U) SYSTEM DEVELOPMENT CORP SANTA
MONICA CA 02 DEC 81 SDC-TM-7038/209/00
UNCLASSIFIED DCA100-80-C-0044
F/G 9/2
NL
1/
END
DATE FILMED
4 83
DTIC



SDC
System Development Corporation
2500 Colorado Avenue, Santa Monica, CA 90406, Telephone (213) 820-4111

series base no./vol./reissue
TM- 7038/209/00
author
DTI Staff
technical
Carl M. Switzky
release
David J. Kaufman
for
Charles A. Savant
date
12/2/81

TM a working paper

This document was produced by
System Development Corporation in performance of Contract DCA100-80-C-0044.

DCEC PROTOCOLS STANDARDIZATION PROGRAM

PROPOSED DoD FTP STANDARD

ABSTRACT

A file transfer service provides for transferring files between network hosts. This document, prepared by Digital Technology Inc. under subcontract to System Development Corporation, specifies a file transfer service for Department of Defense use. The file transfer service is organized as three categories of services: (1) Basic Services, (2) File Protocol Services, and (3) Virtual File System Services. This organization increases the flexibility of the file transfer service. Service primitives that implement the file transfer service are also defined. Finally, the document also describes the services that the file transfer service requires from lower network service layers.

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER 7038/209/00	2. GOVT ACCESSION NO. AD-A126561	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Proposed DoD FTP Standard		5. TYPE OF REPORT & PERIOD COVERED interim technical report 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) DTI Staff		8. CONTRACT OR GRANT NUMBER(s) DCA100-82-C-0044
9. PERFORMING ORGANIZATION NAME AND ADDRESS System Development Corporation 2500 Colorado Ave. Santa Monica, CA 90406		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS P.E. 33126K Task 1053.558
11. CONTROLLING OFFICE NAME AND ADDRESS Defense Communications Engineering Center Switched Networks Engineering Directorate 1860 Wiehle Ave. Reston, VA 22090		12. REPORT DATE 2 Dec 81 13. NUMBER OF PAGES 55
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION DOWNGRADING SCHEDULE N/A

16. DISTRIBUTION STATEMENT (of this Report)

Approved for Public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

N/A

18. SUPPLEMENTARY NOTES

This document represents results of interim studies which are continuing at the DCEC of DCA.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Protocols, Data Communications, Data Networks, Protocol Standardization, File Transfer Protocols

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

A file transfer service provides for transferring files between network hosts. This document, prepared by Digital Technology Inc. under subcontract to System Development Corporation, specifies a file transfer service for Department of Defense use. The file transfer service is organized as three categories of services; (1) Basic Services, (2) File Protocol Services, and (3) Virtual File System Services. This organization increases the flexibility of the file transfer service. Service primitives that implement the file transfer service are also defined. Finally, the document also describes the services that the file transfer service requires from lower network service layers.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

CONTENTS

LIST OF FIGURES

1. OVERVIEW

1.1 INTRODUCTION

This document specifies services for transferring files between computer systems. A major difficulty in defining such services is that network hosts may employ different conventions to access and represent files. This problem is resolved by introducing a virtual file system (VFS) model, which provides a common view of dissimilar local file systems.

The virtual file system model provides a unified basis and terminology for defining a general file transfer service (FTS). This service resides in the presentation layer of the models described by ISO [ISO, 1981a] and SYTEK [SYTEK, 1981]. The service is provided by the software modules that implement a file transfer protocol (FTP). An FTP uses session layer services to provide the file transfer service to the application layer.

This document is divided into eight major sections.

a. Section 1 contains introductory material.

b. Section 2 describes a virtual file system, which defines the common interface between dissimilar file systems.

c. Section 3 describes the file transfer services provided to the application layer.

d. Section 4 specifies the service interface to the application layer.

e. Section 5 describes the services required from the session layer.

f. Section 6 specifies the service interface with the session layer.

g. Section 7 specifies the protocol.

h. Section 8 specifies the services required from the local system environment.

Sections 1, 2, 3, and 5 are provided in this initial version. Service primitives are described in Section 4 to better illustrate the services described in Section 3 and to indicate a possible direction for later specification efforts.

1.2 BACKGROUND

The ability to transfer files between computer systems is one of the basic presentation layer services of a network. A file transfer protocol is primarily concerned with three kinds of services:

a. virtual file system services - the common representation of file structure and data

b. file transfer services - the reliable access and transfer of file data between systems

c. file management services - the management of files in a distributed environment

Most of the work done over the last ten years has focused on the first two of these functions. Only cursory attention has been given to file management, and as yet no concerted effort has been made to design a file management service.

The earliest FTP was developed for the ARPANET by the Network Working Group [Neigus, 1973]. This protocol provided for the transfer of complete files. The files could be structured either as a stream of bytes or as records. The protocol specified common representations only for binary and text data. The transfer mechanisms allowed for checkpoint recovery, data compression, and control of a transfer between two hosts from a third host. The protocol also included several file management services to delete and rename files, and to list the contents of directories.

Subsequent work by other researchers relied heavily on the ARPANET FTP. Three major innovations resulted from this subsequent work. First, it was recognized that the FTP should be partitioned into separate file transfer and file management services. In addition the file transfer service was generalized to allow the transfer of parts of files [Day, 1973]. Second, the FTP for the Experimental Packet Switched System (EPSS) in the United Kingdom introduced the concept of a checkpoint window to limit the number of restart markers that could be sent without an acknowledgement [EPSS, 1975]. Third, the common forms of file structure and data type were greatly expanded (e.g., standard representations of numbers, indexed files) [Schicker, Duenki, Baechi, 1975; Heinze and Butscher, 1977; Forsdick and McKenzie, 1979].

In 1978, an FTP was developed for the Exploratory Data Network (EDN) of the Department of Defense (DoD). This protocol brought together much of the thinking of the previous six years. In the last two years there has been considerable activity among standards bodies directed toward producing an FTP standard. The focus of this effort has been the work of the ISO Open System Interconnection group (ISO TC97/SC16) with major inputs from the National Bureau of Standards (NBS) and the DoD in the U.S., the European Computer Manufacturers Association, as well as national contributions from France, the United Kingdom, and Germany. However, the work has progressed slowly. The ISO group is still refining the virtual file service model and has yet to produce a draft service specification for the basic transfer service. NBS is producing its own FTP that appears to be based on the EDN work. However, only a service specification exists as of this writing.

There appears to be little difference in the basic direction of the EDN, NBS, and ISO work. Though the style and the precise facilities provided differ, no fundamental differences are apparent at the current level of detail. Therefore, since the EDN FTP is the most concretely specified of the FTPs, much of this document is based on that work [Forsdick and McKenzie, 1979].

1.3 OVERVIEW OF THE FILE TRANSFER SERVICE

The file transfer service (FTS) provides the services necessary to move files between network hosts. The ordinary file copy operation which copies data between files at a single host becomes greatly complicated when generalized to a multi-host environment. In addition to the problem of coordinating communication between network hosts, the hosts may employ different conventions to access and represent files. Therefore, the FTS provides a variety of services to manage connections as well as services to provide a common transfer syntax for transferring files with special structure.

The file transfer service defined here is organized into three categories of services:

a. Basic Services consist of a minimal set of services for moving files from one host to another with little consideration for data format. This category includes services that allow the user to

   1. open connections to foreign hosts
   2. negotiate extensions to the Basic Services
   3. transmit unstructured files
   4. abort transfers and terminate connections
   5. request and receive status information on a transfer

b. File Protocol Services extend the Basic Services by providing extra facilities for manipulating connections and data transmission. This category includes services that allow the user to

   1. conduct partial file transfers
   2. conduct three-party transfers
   3. use security parameters in the file selection process
   4. request data compression during a transfer
   5. request flow control services

c. Virtual File System Services extend the Basic Services by providing facilities for transmitting files with special structures or data types. This category includes services that allow the user to

   1. transmit record structured files
   2. transmit indexed files
   3. transmit binary files

4. transmit integer files

5. transmit floating point files

Given the above organization, the Basic Services require few resources to implement and use. This encourages partial implementations. Extended services can be added incrementally by implementing the appropriate service primitives.

1.4 SCENARIO

The following scenario illustrates the use of the FTS in transferring a file. This scenario takes a service oriented point of view. That is, the transfer is seen as an interaction between a service user and the FTS.

a. Connection Establishment Phase When the user wishes to transfer a file, he must establish network connections with the hosts where the source and destination files reside. The user makes separate requests to the FTS to establish the necessary connections.

   The participating hosts request login information to complete the connection establishment. The FTS obtains this information from the user and passes it along to the appropriate host.

   The FTS notifies the user when each login is successfully completed.

b. Negotiation Phase Next the user initiates the negotiation of the options needed for the intended transfer. The user makes separate requests to negotiate options with the participating hosts. If, for example, the options required for this example transfer are the capabilities to support three-party transfers, record structured file transfers, and binary file transfers, then the FTS separately negotiates these options with the participating hosts and notifies the user that the negotiations were successful.

c. File Selection Phase When negotiations are complete, the user employs file selection primitives to inform the FTS of the characteristics of the source and destination files. This information identifies the source and destination files and specifies their structure as, for example, a binary file with fixed length records. The FTS passes this information to the participating hosts.

d. Data Transfer Phase When source and destination files have been selected, the user specifies the data transfer operation to the FTS. In this scenario the entire file is to be transferred, and the destination host is to store the file (rather than append or retrieve it).

   The FTS passes this information to the participating hosts, and they perform the transfer in accordance with the characteristics specified in the file selection phase.

The FTS informs the user when the transfer is complete. The user may now either request another transfer or terminate the session.

e. Termination Phase With the completion of the transfer, the user asks the FTS to terminate the session.

After dissolving the connections associated with the session, the FTS informs the user of its successful termination. The transfer session is complete.

2. VIRTUAL FILE SYSTEM MODEL

2.1 OVERVIEW

The VFS represents an ideal file system that supports most of the features found in real file systems. The VFS defines a file as a container of data that has file attributes that describe the file and its contents. The VFS file attributes have been chosen so that real file attributes can be mapped easily onto their VFS counterparts.

A major restriction of the VFS defined here is that VFS files can only contain data of a single data type. An extension to include files with multiple data types might be useful, but its inclusion would substantially complicate the definition of the file transfer service. Therefore, the inclusion of such an extension is reserved for future study.

The VFS provides a basis for defining the file transfer service. The FTS allows a user to select source and destination files in terms of VFS attributes (e.g., file name and file data type). In order to transfer a file, the FTS provides a transfer syntax that preserves the file structures and data types defined for VFS files. The source host maps the file data from its local syntax into the transfer syntax defined by the FTS. Upon receipt of this data, the destination host maps the data from the transfer syntax into its own local syntax. This approach requires only that hosts be able to map back and forth between their local syntax and the transfer syntax defined for VFS files. It is not necessary to implement mappings between each pair of dissimilar hosts. The relationships between the local host file systems, the VFS, and the FTS are summarized in Figure 1.

2.2 VIRTUAL FILE SYSTEM ATTRIBUTES

The VFS may be described in terms of the following file attributes.

2.2.1 General Attributes

a. FileName. This attribute uniquely identifies a file at the local host where it resides. At the present time there is no network-wide system of naming files. Therefore, the choice of file names is strictly a local concern.

b. FilePassword. This attribute gives the password used to control access to a file at the local host. Not all hosts will support this attribute, and not all files will possess this attribute even at hosts that do support the attribute.

c. SecurityLevel. This attribute identifies the security level of a file. Not all hosts will support this attribute, and not all files will possess this attribute even at hosts that do support the attribute.

d. TCC. This attribute gives the Transmission Control Code for a file. Not all hosts will support this attribute, and not all files will possess this attribute even at hosts that do support the attribute.

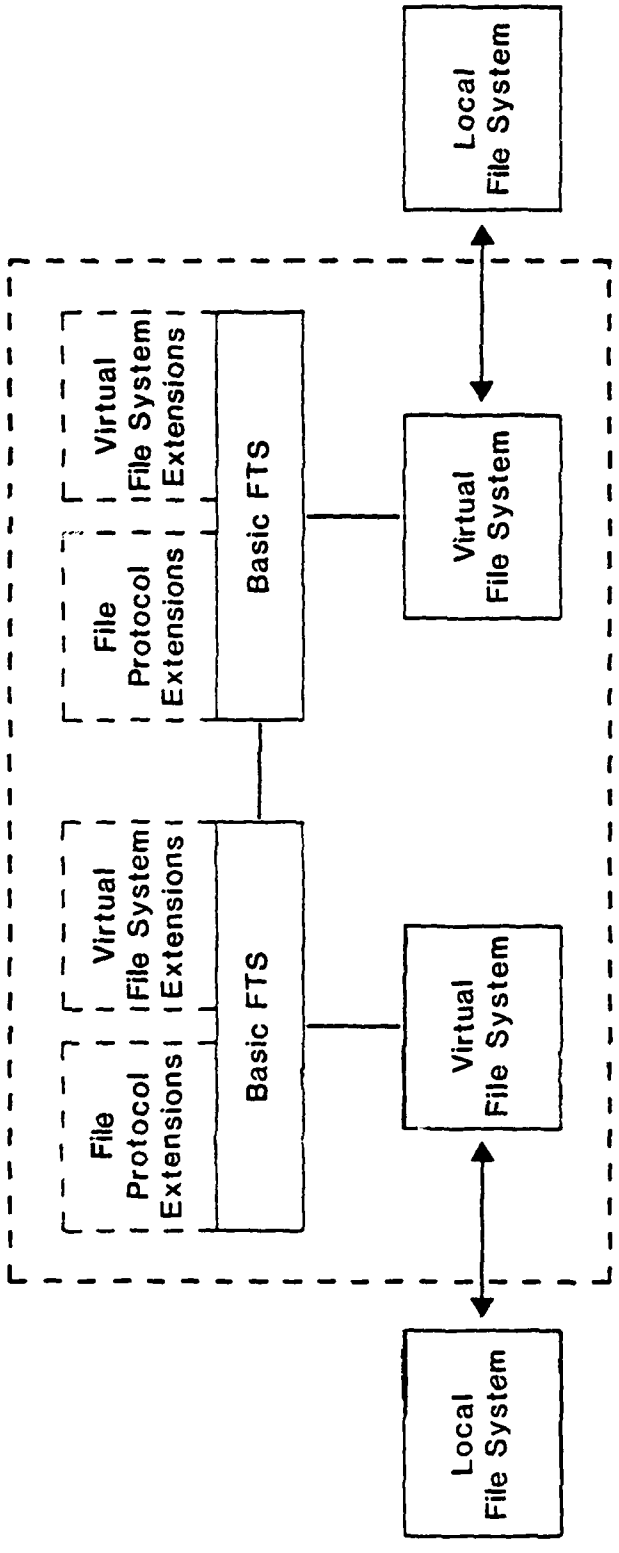


Figure 1. The Virtual File System Model

e. FileLength. This attribute gives the approximate length of a file in octets. Since local file systems may represent data differently, only an approximation can be given. The attribute is used to give a rough idea of the size of a file.

f. FileStructure. This attribute describes the organization of data items in a file. In this model we assume that a record is a sequence of data items of a given type. As noted above, we do not allow for records consisting of fields of more than one data type. The FileStructure attribute may take one of the following values:

1. Unstructured, data items in a sequence
2. Record structured, fixed record length
3. Record structured, variable record length
4. Record structured, fixed record length, and indexed
5. Record structured, variable record length, and indexed

g. FileDataType. This attribute describes the kind of atomic data items that may appear in a file. The FileData attribute may take one of the following values:

1. Text
2. Binary
3. Integer
4. Floating point

For each of the above data types, additional attributes are needed to fully characterize the representation of data in a file. These are summarized by data type below.

2.2.2 Text File Attributes

Specifying the representation of data in text files is complicated by the variety of formatting conventions used in these files. Before giving the attributes that specify text file representation, we present a short discussion of these conventions. The VFS recognizes two main classes of formatting conventions, the "record" class and the "stream" class.

The record class treats each line of text as a record. Horizontal spacing is achieved only by using blank characters. The end of the line is implicit in the end of the record. Vertical spacing is specified using one of the ANS FORTRAN carriage control characters as the first character in a line. The VFS recognizes the following line spacing options:

1. Single spacing

2. Double spacing

3. To first line of next page

4. No spacing

The stream class treats a text file as a stream of characters with embedded format effector characters. The format effector characters mark the end of lines and specify horizontal and vertical spacing. The VFS recognizes the following format effectors:

1. HorizontalTab

2. VerticalTab

3. BackSpace

4. CarriageReturn

5. LineFeed

6. NewLine

7. FormFeed

The following attributes use the above formatting conventions to characterize the representation of data in a VFS text file:

FormatClass. This attribute identifies the class of formatting conventions used in a text file. This attribute may take the following values:

1. Record1 - The file consists of records; each record is a line of text. No formatting information is provided either by carriage control characters or by format effectors.

2. Record2 - The file consists of records; each record is a line of text. The first character of each line is one of the first three carriage control characters listed above. The "no spacing" carriage control option may not be used.

3. Record3 - The file consists of records; each record is a line of text. The first character of each line is one of the four carriage control characters listed above.

4. Stream - The text formatting is entirely determined by embedded format effectors.

FormatEffectors. This attribute lists the format effector options that are used in a text file. This attribute is irrelevant unless the FormatClass attribute is set to the Stream option defined above. The

following options may appear in the FormatEffectors list:

1. HorizontalTab
2. VerticalTab
3. BackSpace
4. CarriageReturn
5. LineFeed
6. CarriageReturn implies line feed action as well
7. LineFeed implies carriage return action as well
8. NewLine
9. FormFeed

HorizontalTabStops. This attribute lists the locations for the horizontal tab stops in a file.

VerticalTabStops. This attribute lists the locations for the vertical tab stops in a file.

CharacterCode. This attribute identifies the system of encoding characters in a file. This attribute may take the following values:

1. ASCII
2. EBCDIC

Parity. This attribute specifies the parity convention used to represent characters in a file. This attribute is irrelevant if the CharacterCode attribute is set to EBCDIC. This attribute may take the following values:

1. Parity bit zero
2. Even parity
3. Odd parity
4. Any parity

2.2.3 Binary File Attributes

The following attribute completely characterizes the representation of data in a binary file:

BinaryLength specifies the number of binary digits in a word of data.

2.2.4 Integer File Attributes

Integer files consist of words of data representing twos complement integers. The following attribute completely characterizes the representation of data in an integer file:

IntegerLength specifies the bit length of each integer.

2.2.5 Floating Point File Attributes

Floating point files consist of words of data representing floating point numbers. Floating point data is represented according to the proposed IEEE standard [IEEE, 1981]. The following attribute completely characterizes the representation of data in a floating point file:

FloatingPointLength specifies whether the data is single or double precision. (Consideration of a parameter to represent various word formats and radices requires further study. The IEEE Task 854 working group is currently developing a floating point standard that would allow such parameterization.)

2.3 ADDITIONAL VIRTUAL FILE SYSTEM ATTRIBUTES

In addition to the attributes defined above, several other potentially useful attributes can be identified. Although the following attributes will not be used in the definitions of the service primitives, they may be useful in defining service extensions. Therefore, they are included for completeness. A more complete discussion of these attributes may be found in the ISO file service description [ISO, 1980].

a. AccessControl. This attribute lists the entities that may access a file and gives the operations that may be performed by each entity.

b. LegalRequirements. This attribute describes a file in terms of the legal ownership of the data in the file. This information includes the legal owner and the license or registration number.

c. AccountingAndCharging. This attribute describes a file with respect to accounting charges for file access and storage costs.

d. History. This attribute lists information regarding the history of events associated with a file, including the date of the event and the entity responsible for the event. Typical events include the last access, the last modification, and creation of the file.

e. GradeOfStorage. This attribute describes characteristics of the storage provided to a file by the local file system.

3. SERVICES PROVIDED TO THE UPPER LAYER

3.1 INTRODUCTION

The file transfer service is one of the services provided by the presentation layer of the models described by ISO [ISO, 1981a] and SYTEK [SYTEK, 1981]. The FTS uses the services provided by the session layer to provide file transfer services to the application layer.

The FTS allows users to establish and manage file transfer connections in order to transfer structured files between hosts. The FTS is described in more detail in Section 3.4 below. First, however, the connection architecture and the phases of operation must be described to provide a general context for the description of the FTS.

3.2 FILE TRANSFER SERVICE ARCHITECTURE

When files are copied between hosts, roles for three hosts can be identified:

1. The controller host, where the user resides
2. The donor host, where the source file resides
3. The recipient host, where the destination file resides

The user communicates with a controller FTS module to specify the desired transfer services. The controller FTS module specifies the transfer to both the donor and the recipient and reports the status of the transfer to the user. The donor and recipient hosts actually transfer the file (see Figure 2).

Usually the controller host also acts as either donor or recipient. Therefore the file transfer service is defined as a two-party system, but the three-party case is included as an extension.

3.3 PHASES OF OPERATION

An FTS session is divided into several phases of operation that are distinguished by the type of operations performed. Services are classified by the phase in which they are performed. The phases are as follows:

1. Connection establishment phase
2. Negotiation phase
3. File selection phase
4. Data transfer phase
5. Termination phase

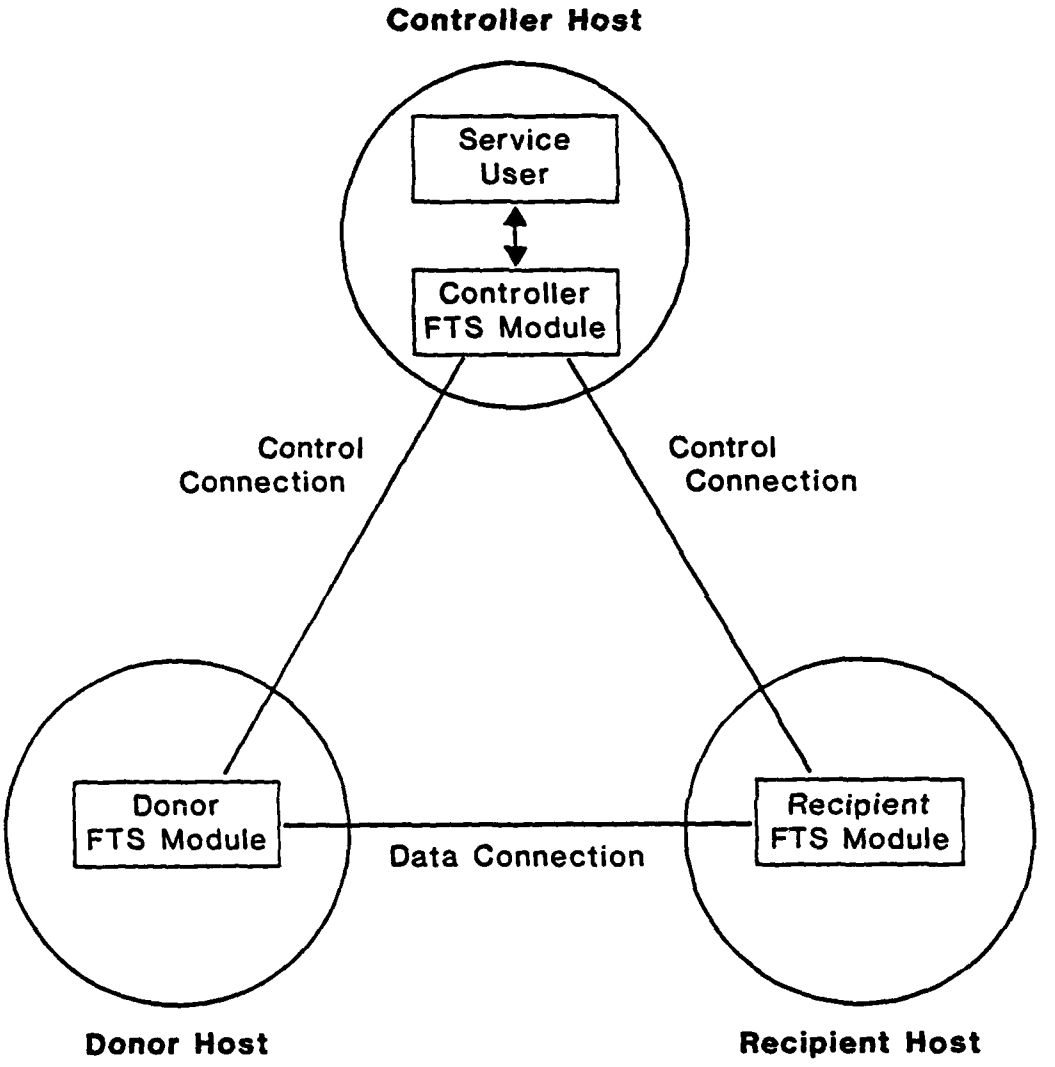


Figure 2. File Transfer Service Architecture

In the first three phases, the user makes requests to establish connections with the donor and recipient hosts, to negotiate options with them, and to select the source and destination files. The donor and recipient can then transfer the file. The session can then enter the termination phase or return to an earlier phase to perform other file transfer tasks. In greater detail, the phases proceed as follows:

3.3.1 Connection Establishment Phase

The user instructs the controller to open separate connections to the hosts acting as donor and recipient in the transfer. The foreign hosts request login information and the controller supplies them with user name, account, password, security level, and Transmission Control Code (TCC) from the user.

3.3.2 Negotiation Phase

The user instructs the controller to conduct option negotiation with the donor and recipient.

3.3.3 File Selection Phase

The user instructs the controller to send file selection information to each of the hosts to establish parameters for the transfer. This includes the VFS attributes necessary to define the source and destination files and additional information to define the mode of transfer.

3.3.4 Data Transfer Phase

The user instructs the FTS to send files or portions of files. The controller relays this information to the donor and recipient hosts, and they transfer the data. The FTS notifies the user upon completion of data transfer.

3.3.5 Termination Phase

When no file transfer is in progress, the user can instruct the controller to terminate a session. If a file transfer is in progress, the user must first request that the transfer be aborted.

When a file transfer is complete, it is not necessary to enter the termination phase. The session can enter the termination phase or previous phases as follows:

1. The user can request that the session be terminated.
2. The user can request the transfer of another section of the file.
3. The user can reenter the file selection process to prepare for transferring a different file.
4. The user can request that the controller renegotiate options.

Figure 3 illustrates the principal phase transitions. Transitions to the termination phase are not shown since a session can be terminated from any phase.

3.4 DESCRIPTION OF SERVICES

3.4.1 Basic Services

The Basic Services consist of a minimal set of services required to move files from one host to another with little consideration for data format. These services are classified as follows.

3.4.1.1 Connection Establishment

This service provides for the establishment of connections to hosts specified by the user. The user can also specify quality of service parameters including transit delay, setup delay, and error rate. These parameters are passed directly to the session layer to establish a connection with the desired attributes. Other quality of service parameters may be added in the future. The FTS provides a unique name, the TransactionID, with which the user refers to the connections belonging to a given session.

To complete a connection establishment, a foreign host requires authentication data from the user. The FTS identifies a user to a foreign host by giving the user's account, user name, password, security level, and TCC. This service may involve a general network authentication service.

3.4.1.2 Negotiation

This service allows the user to negotiate the use of services beyond the basic services. The main extensions belong to the File Protocol Services and the Virtual File System Services discussed above. New services can also be added (see Appendix A). The currently defined extensions may be summarized as follows:

a. PartialTransfer. This service allows parts of files to be transferred.

b. ThreeParty. This service allows the use of commands to specify three-party transfers.

c. Security. This service allows the use of commands to specify the password, security level, and TCC of the file during the file selection phase.

d. FlowControl. This service allows the use of the FTS flow control service.

e. Compression. This service allows the use of data compression in file transfers.

f. SourceFileDescription. This service provides a complete description of the characteristics of the selected file to be given to the user before a file transfer.

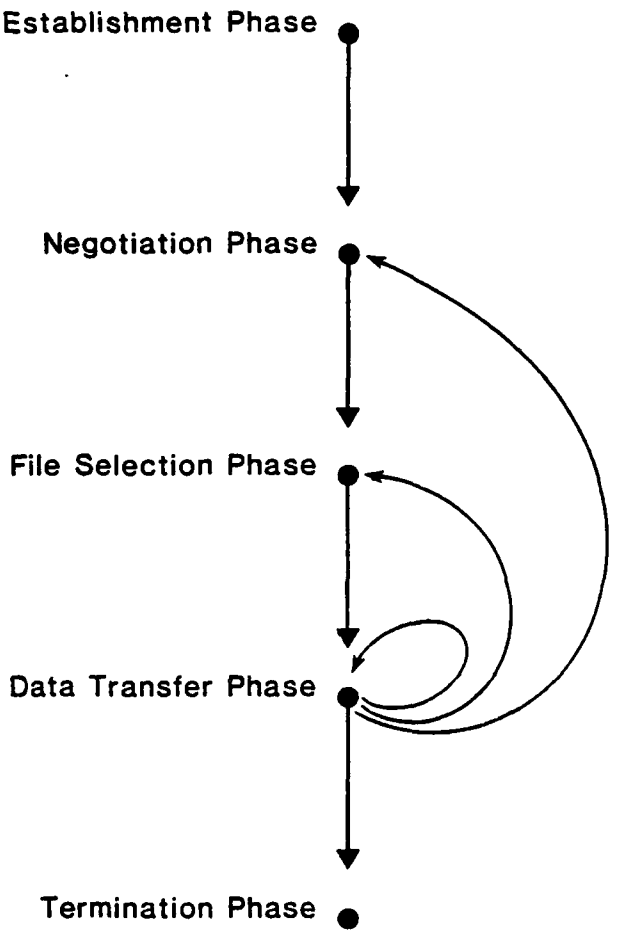


Figure 3. Phase Transitions

g. FileParameters. This service allows the structure of a file to be specified during the file selection phase.

h. Record. This service allows the use of primitives and parameters to specify the transfer of record structured files.

i. Indexed. This service allows the use of primitives and parameters to specify the transfer of indexed files.

j. Text. This service allows the use of primitives and parameters to specify the transfer of text files.

k. Binary. This service allows the use of primitives and parameters to specify the transfer of binary files.

l. Integer. This service allows the use of primitives and parameters to specify the transfer of integer files.

m. FloatingPoint. This service allows the use of primitives and parameters to specify the transfer of floating point files.

3.4.1.3 File Selection

This service allows the user to select source and destination files for the transfer.

3.4.1.4 File Transfer

This service provides the ability to copy a file from one file system to another. The recipient may append the file onto an existing file or store it in a new file. The user is notified upon completion of the transfer.

3.4.1.5 Termination

These services allow the user to terminate a file transfer. The AbortTransfer Service terminates a file transfer without terminating the FTS connections. The Disconnect Service closes the connections, and terminates the transfer session.

3.4.1.6 Transfer Status

This service allows the user to inquire and be informed of the status of a transfer session. Status information includes host status, transaction status, and a summary message in text form for display to a human user. This service is not phase oriented since it may occur in any phase.

3.4.2 File Protocol Services

The File Protocol Services provide the following extra capabilities for managing file transfers:

a. Partial Transfers. This service allows the transfer of parts of a file.

b. Three-party Transfers. This service allows the user to transfer a file when the donor and recipient hosts are both foreign hosts.

c. File Security. This service allows the user to specify the security level, TCC, and file password for a transferred file.

d. File Compression. This service allows the user to specify that data compression be used in a file transfer. If this option is successfully negotiated, the FTS may independently decide to use compression without a request from the user.

e. Flow Control. This service allows the user to halt and resume a file transfer. This is intended to be used when a transfer is halted for a reasonably long time, for example, to change a tape or to put paper in a printer.

f. Source File Description. This service provides the user with a complete description of the selected source file.

g. File/Process Transfers. This service allows a process to replace a file as source or destination in a transfer. This service is reserved for further study.

3.4.3 Virtual File System Services

The Virtual File System services provide for transferring the following kinds of structured and/or typed files:

a. Record Structured Files. This service allows the user to specify the transfer of a file as a record structured file. The transfer syntax for fixed record length or variable record length files will be used as appropriate to preserve the record structure of the file.

b. Indexed Files. This service allows the user to specify the transfer of a record structured file as an indexed file. The transfer syntax for indexed, record structured files will be used to preserve the indexed structure of the file.

c. Text Files. This service allows the user to specify the transfer of a file as a text file. The user may specify the transfer syntax with respect to character encoding and parity; the default syntax is ASCII encoding with any parity.

d. Binary Files. This service allows the user to specify the transfer of a file as a binary file. The user may specify whether the words of data are to be packed end-to-end in the data stream or whether each new word of binary data will begin on a new octet.

e. Integer Files. This service allows the user to specify the transfer of a file as an integer file. The user may specify whether the words of data

are to be packed end-to-end in the data stream or whether each new word of binary data will begin on a new octet.

f. Floating Point Files. This service allows the user to specify the transfer of a file as a floating point file. The user may specify whether the words of data are to be packed end-to-end in the data stream or whether each new word of binary data will begin on a new octet.

4. UPPER LAYER SERVICE/INTERFACE SPECIFICATION

4.1 INTRODUCTION

The service primitives in this section structure the dialogue between the service user and the FTS. It is assumed that the mechanism that provides this communication allows immediate notification of erroneous or malformed primitive invocations. The primitive specifications are intended to clarify the description of the services in the previous section by specifying the units of information exchanged between the FTS and the user. However, these primitive specifications are preliminary and do not fully conform to the System Development Corporation guidelines [Simon, 1981].

4.2 BASIC SERVICES

4.2.1 Establishment Phase Primitives

ConnectRequest
- TransactionID (optional)
- Host
- TransitDelay
- SetupDelay
- ErrorRate

Function

The ConnectRequest primitive is invoked by the user to request the establishment of an FTS connection to a foreign host.

Parameter Semantics

TransactionID

This optional parameter uniquely identifies the current transfer session. The unique identifier is used to bind the requested connection to the already existing session identified by the parameter value. The parameter is used in the establishment of three-party transfer sessions. In this case, the user first opens a connection to one of the participating hosts by invoking the primitive without using the TransactionID parameter. If the invocation is successful, the ConnectResponse primitive passes back the TransactionID value corresponding to the new connection. Then, the user invokes the ConnectRequest primitive a second time using the given TransactionID value. The FTS then associates the second connection with the same TransactionID value as the first connection.

Host

This parameter identifies the host to which the connection is to be established.

TransitDelay

This parameter establishes the quality of service with respect to throughput transit delay. Like the other quality of service parameters below, this parameter will ultimately be passed directly to the

transport layer.

SetupDelay
This parameter establishes the quality of service with respect to connection set up delay. Like the other quality of service parameters, this parameter will ultimately be passed directly to the transport layer.

ErrorRate
This parameter establishes the quality of service with respect to residual error rate. Like the other quality of service parameters, this parameter will ultimately be passed directly to the transport layer.

When Generated
This primitive may be invoked whenever the user desires to establish an FTS connection.

Effect of Receipt
The FTS attempts to open a connection as specified by the primitive invocation. The results of this attempt are returned to the user by means of the ConnectResponse primitive.

ConnectResponse
TransactionID
ConnectStatus

Function
The ConnectResponse primitive is invoked by the FTS to inform the user of the result of a ConnectRequest primitive invocation.

Parameter Semantics

TransactionID
This parameter identifies the session in which the connection resides, if the establishment was successful.

ConnectStatus
This parameter indicates whether the connection attempt succeeded or gives the reason for its failure.

When Generated
The ConnectResponse primitive is generated by the FTS after receiving a ConnectRequest primitive and attempting to open the requested connection.

Effect of Receipt If the connection attempt succeeded, the user saves the TransactionID value for later reference to the given transfer session. Otherwise, no action is required.

LoginRequest
    TransactionID
    Host

Function
The LoginRequest primitive is invoked by the FTS to inform the user that the given host has requested login information.

Parameter Semantics

TransactionID
    This parameter identifies the session in which the connection resides.

Host
    This parameter identifies the host that requested the authentication data.

When Generated
The LoginRequest primitive is generated by the FTS when it receives a request from a foreign host for authentication data from the user.

Effect of Receipt
The user should reply with a LoginInfo primitive invocation supplying the requested data.

LoginInfo

TransactionID
Host
UserName
Account
Password
SecurityLevel
TCC

Function

The LoginInfo primitive is invoked by the user to supply the authentication data requested by the given foreign host via a LoginRequest primitive invocation.

Parameter Semantics

TransactionID
This parameter identifies the session in which the connection resides.

Host
This parameter identifies the host for which the authentication data is intended.

UserName
This parameter identifies the service user to the foreign host.

Account
This parameter identifies the account to which the session will be charged.

Password
This parameter gives the user's password for the given account on the foreign host.

SecurityLevel
This parameter gives the security level to be used by the user during the current transfer session.

TCC
This parameter gives the Transmission Control Code to be used by the user during the current transfer session.

When Generated

The LoginInfo primitive is generated by the user in response to a LoginRequest primitive requesting authentication data to complete the establishment of a connection.

Effect of Receipt

The FTS passes the authentication data along to the given host.

4.2.2 Negotiation Phase Primitives

RequestRange
Host

Function
The RequestRange primitive is invoked by the user to request information concerning the options supported by a given host.

Parameter Semantics

Host
This parameter identifies the host from which the option information is requested.

When Generated
The user may generate a RequestRange primitive whenever there is an open connection to the given host.

Effect of Receipt
The FTS obtains the information from the given host and returns that information to the user in an Indicate primitive.

Indicate

Host
NewParameter (optional, may be repeated)
NewCommand (optional, may be repeated)
NewClass (optional, may be repeated)

Function

The Indicate primitive is invoked by the FTS to inform the user of the options supported by a given host. The currently defined options are the extensions listed in Section 3.4.1.2. A mechanism for defining new extensions is discussed in Appendix A.

Parameter Semantics

Host

This parameter identifies the host that offers the options specified by the other parameters.

NewParameter

This parameter indicates that the host supports the protocol extension consisting of the use of the given new parameter. The value of NewParameter is the parameter index of the new parameter. NewParameter is optional and may be repeated as many times as necessary to list all of the parameter extensions supported by the given host.

NewCommand

This parameter indicates that the host supports the protocol extension consisting of the use of the given new command. The value of NewCommand is the command index of the new command. NewCommand is optional and may be repeated as many times as necessary to list all of the command extensions supported by the given host.

NewClass

This parameter indicates that the host supports the protocol extension consisting of the use of the given new class. The value of NewClass is the class index of the new class. NewClass is optional and may be repeated as many times as necessary to list all of the class extensions supported by the given host.

When Generated

The FTS invokes the Indicate primitive when the FTS has received the information requested by an earlier RequestRange primitive.

Effect of Receipt

No action is required by the user. However, the user will probably use the information to determine what parameter values to use in a later invocation of the Set primitive.

Set

TransactionID
Host
NewParameter (optional, may be repeated)
NewCommand (optional, may be repeated)
NewClass (optional, may be repeated)

Function

The Set primitive is invoked by the user to instruct the FTS to negotiate with the given host to set the specified options. The currently defined options are the extensions listed in Section 3.4.1.2. A mechanism for defining new extensions is discussed in Appendix A.

Parameter Semantics

TransactionID
This parameter identifies the current transfer session.

Host
This parameter identifies the host with which the options are to be negotiated.

NewParameter
This parameter indicates that the FTS should attempt to set the option allowing the use of the given new parameter. The value of NewParameter is the parameter index of the new parameter. NewParameter is optional and may be repeated as many times as necessary to list all of the parameter extensions that the FTS should attempt to set.

NewCommand
This parameter indicates that the FTS should attempt to set the option allowing the use of the given new command. The value of NewCommand is the command index of the new command. NewCommand is optional and may be repeated as many times as necessary to list all of the command extensions that the FTS should attempt to set.

NewClass
This parameter indicates that the FTS should attempt to set the option allowing the use of the given new class. The value of NewClass is the class index of the new class. NewClass is optional and may be repeated as many times as necessary to list all of the class extensions that the FTS should attempt to set.

When Generated

The Set primitive may be generated by the user whenever there is an open connection to the given host.

Effect of Receipt

The FTS attempts to set the options given by the parameters. When the negotiation is complete, the FTS informs the user of the results with a SetResponse invocation.

SetResponse

TransactionID
Host
NewParameter (optional, may be repeated)
NewCommand (optional, may be repeated)
NewClass (optional, may be repeated)

Function

The SetResponse primitive is invoked by the FTS to inform the user of the results of a Set primitive invocation.

Parameter Semantics

TransactionID

This parameter identifies the current transfer session.

Host

This parameter identifies the host with which the options were negotiated.

NewParameter

This parameter indicates that the given new parameter may now be used with the given host. The value of NewParameter is the parameter index of the new parameter. NewParameter is optional and may be repeated as many times as necessary to list all of the parameter extensions that may now be used.

NewCommand

This parameter indicates that the given new command may now be used with the given host. The value of NewCommand is the command index of the new command. NewCommand is optional and may be repeated as many times as necessary to list all of the command extensions that may now be used.

NewClass

This parameter indicates that the given new class may now be used with the given host. The value of NewClass is the class index of the new class. NewClass is optional and may be repeated as many times as necessary to list all of the class extensions that may now be used.

When Generated

The SetResponse primitive is generated by the FTS upon completion of option negotiation with the given host.

Effect of Receipt

No action by the user is required. However, the results of the option negotiation determine what actions may subsequently be taken by the user.

4.2.3 File Selection Phase Primitives

TransactionDefinition
    TransactionID
    Host
    FileName

Function
The TransactionDefinition primitive is invoked by the user to select the file to be accessed during the transfer session.

Parameter Semantics

TransactionID
    This parameter identifies the current transfer session.

Host
    This parameter identifies the host at which the given file is to be accessed.

FileName
    This parameter gives the name of the file to be accessed.

When Generated
The TransactionDefinition primitive may be invoked by the user whenever a connection to the named host is open and no transfer is in progress.

Effect of Receipt
The FTS uses the parameter values to select a file at the named host.

4.2.4 Data Transfer Phase Primitives

BeginTransfer
TransactionID
Donor
Recipient
Action
Start
End

Function
The BeginTransfer primitive is invoked by the user to instruct the FTS to initiate the transfer of data. The invocation of this primitive signals the end of the file selection phase and the start of the data transfer phase.

Parameter Semantics

TransactionID
This parameter identifies the current session.

Donor
This parameter identifies the donor host in the specified transfer. Unless the ThreeParty option has been successfully negotiated, either the donor or recipient must be the controller host.

Recipient
This parameter identifies the recipient host in the specified transfer. Unless the ThreeParty option has been successfully negotiated, either the donor or recipient must be the controller host.

Action
This parameter specifies the action to be taken by the recipient host. The values for this parameter are:

Store
Append

Start
This parameter identifies the location in the file where the transfer is to begin. The units of this parameter are as follows:

Record files:	records
Indexed files:	keys
Text files:	lines
Binary files:	words
Integer files:	words
Floating point files:	words

This parameter will be implemented as a 32-bit quantity. If the file is an indexed file, the parameter is interpreted as a left justified bit string which specifies the key where the transfer is to begin. Otherwise, the parameter is an unsigned integer which

specifies the number of the first unit of data to transfer.

Unless the PartialTransfer option has been successfully negotiated, this parameter must specify the beginning of the file.

End

This parameter identifies the location in the file where the transfer is to end. The data type and units of this parameter are the same as those of the Start parameter. The end of the file is specified by setting all the bits of the parameter.

Unless the PartialTransfer option has been successfully negotiated, this parameter must specify the end of the file.

When Generated

The BeginTransfer primitive is generated by the user when the file selection process is complete.

Effect of Receipt

The FTS initiates the data transfer specified by the parameters.

EndOfFile
TransactionID

Function
This primitive is invoked by the FTS to inform the user that the current transfer has ended.

Parameter Semantics

TransactionID
This parameter identifies the current session.

When Generated
The EndOfFile primitive is generated upon completion of a file transfer.

Effect of Receipt
No action is required.

4.2.5 Termination Phase Primitives

AbortTransfer
    TransactionID

Function
The AbortTransfer primitive is invoked by the user to request that the current transfer be terminated.

Parameter Semantics

    TransactionID
        This parameter identifies the current session.

When Generated
An AbortTransfer primitive can be generated whenever the TransactionID parameter refers to an existing session.

Effect of Receipt
If a data transfer is in progress, the FTS causes its orderly termination.

DisconnectRequest
TransactionID

Function
The DisconnectRequest primitive is invoked by the user to request the orderly termination of the transfer session.

Parameter Semantics

TransactionID
This parameter identifies the current transfer session.

When Generated
The DisconnectRequest primitive may be invoked whenever the TransactionID parameter refers to an existing session.

Effect of Receipt
The FTS causes an orderly termination of all the connections associated with the given session.

DisconnectResponse
TransactionID
DisconnectCode

Function
The DisconnectResponse primitive is invoked by the FTS to inform the user of the results of a DisconnectRequest.

Parameter Semantics

TransactionID
This parameter identifies the current transfer session.

DisconnectCode
This parameter reports the results of the Disconnect action.

When Generated
The DisconnectResponse primitive is invoked by the FTS after receiving a DisconnectRequest and taking action to terminate the transfer session.

Effect of Receipt
No action is required.

4.2.6 Status Primitives

StatusRequest
    TransactionID
    Host

Function
The StatusRequest primitive is invoked by the user to ask for the status of the host named by the Host parameter.

Parameter Semantics

TransactionID
    This parameter identifies the session for which the status information is requested.

Host
    This parameter identifies the host for which the status information is requested.

When Generated
The StatusRequest primitive may be invoked whenever the parameters refer to an existing session connection.

Effect of Receipt
The FTS attempts to obtain the requested information and return it with the *StatusResponse primitive.*

StatusResponse
TransactionID
Host
HostStatus
TransactionStatus
SystemMessage

Function
The StatusResponse primitive is invoked by the FTS to inform the user of the results of a StatusRequest primitive invocation.

Parameter Semantics

TransactionID
This parameter identifies the session to which the status information applies.

Host
This parameter identifies the host to which the status information applies.

HostStatus
The HostStatus parameter indicates the status of the host:

Running
Awaiting action by controller
Awaiting action by third host

TransactionStatus
The TransactionStatus parameter indicates the status of the current transaction:

Completed
In progress - running
In progress - suspended
Pending (not yet started)
Failed

SystemMessage
This parameter contains text data about session status that is intended for a human reader.

When Generated
The StatusResponse primitive is generated by the FTS when status information requested by the StatusRequest is obtained.

Effect of Receipt
No action by the user is required.

4.3 FILE PROTOCOL SERVICE PRIMITVES

4.3.1 File Selection Phase Primitives

FileSecurity
- TransactionID
- Host
- SecurityLevel
- FilePassword
- TCC

Function
The FileSecurity primitive is invoked by the user to add security information to the description of a file.

Parameter Semantics

TransactionID
This parameter identifies the current transfer session.

Host
This parameter identifies the host where the file to be accessed resides.

SecurityLevel
This parameter identifies the security level of the file to be accessed.

FilePassword
This parameter gives the password of the file to be accessed.

TCC
This parameter gives the Transmission Control Code of the file to be accessed.

When Generated
The FileSecurity primitive may be generated during the file selection phase provided that the Security option has been negotiated with the given host.

Effect of Receipt
The FTS passes the security information to the named host.

FileCompression
TransactionID

Function
The FileCompression primitive is invoked by the user to request that compression be used in the file transfer. The FTS may use compression without receiving this request, but an invocation of this primitive assures that compression will be used.

Parameter Semantics

TransactionID
This parameter identifies the current transfer session.

When Generated
The FileCompression primitive may be invoked during the file selection phase provided that the Compression option has been negotiated with the participating hosts.

Effect of Receipt
If the use of compression has been negotiated, the FTS causes compression to be used in the file transfer.

SourceFileDescription
TransactionID
FileName
FileStructure
FileDataType
FileLength

Function
The SourceFileDescription primitive is invoked by the FTS to provide the user with a full description of the file that has been selected by the donor host.

Parameter Semantics

TransactionID
This parameter identifies the current transfer session.

FileName
This parameter gives the name of the file that has been selected by the file selection commands.

FileStructure
This parameter identifies the structure of the file as:

Unstructured, data items in a sequence
Record structured, fixed record length
Record structured, variable record length
Record structured, fixed record length, and indexed
Record structured, variable record length, and indexed

FileDataType
This parameter identifies the type of the selected file as one of:

Text
Binary
Integer
FloatingPoint

FileLength
This parameter gives the approximate length of the file in octets.

When Generated
The SourceFileDescription primitive may be generated by the FTS following the receipt of the initial BeginTransfer primitive which terminates the file selection phase and begins the transfer phase. This primitive is only employed when the SourceFileDescription option has been negotiated with the donor host.

Effect of Receipt
The user must reply with a SourceFileConfirmation primitive.

SourceFileConfirmation
TransactionID
Agree

Function
The SourceFileConfirmation primitive expresses the user's agreement or disagreement with the choice of a file described by the SourceFileDescription primitive. If the primitive expresses agreement, the transfer can proceed. Otherwise, the user will have to respecify the file before proceeding.

Parameter Semantics

TransactionID
This parameter identifies the current transfer session.

Agree
This boolean parameter is set to true if the user wishes to go ahead with the transfer of the source file described by a previous SourceFileDescription command. Setting this parameter to false blocks the transfer until the BeginTransfer primitive is again invoked.

When Generated
The SourceFileConfirmation primitive may be invoked following the receipt of a SourceFileDescription primitive.

Effect of Receipt
The FTS proceeds with the specified transfer if the Agree parameter is true. Otherwise, no action is taken.

4.3.2 Data Transfer Phase Primitives

Cease

TransactionID
Interval

Function

The Cease primitive is invoked by the user to halt a file transfer for a period of time estimated by the Interval parameter.

Parameter Semantics

TransactionID
This parameter identifies the current transfer session.

Interval
This parameter specifies an estimate of the length of time that the file transfer is to be halted.

When Generated

The Cease primitive may be generated during the data transfer phase provided that the FlowControl option has been negotiated with the participating hosts.

Effect of Receipt

The FTS halts the file transfer until a Resume primitive is received.

Resume
TransactionID

Function
The Resume primitive is invoked by the user to resume a file transfer following its suspension by the Cease primitive.

Parameter Semantics

TransactionID
This parameter identifies the current transfer session.

When Generated
The Resume primitive may be generated by the user when a file transfer has been halted by an invocation of the Cease primitive.

Effect of Receipt
The FTS resumes the file transfer.

4.4 VIRTUAL FILE SYSTEM PRIMITIVES

4.4.1 File Selection Phase Primitives

FileParameters
TransactionID
FileStructure
FileDataType
FileLength

Function
The FileParameters primitive is invoked by the user to establish the structure and data type of the file to be transferred. Depending on the structure and data type of the file, the user will also employ the other virtual file system primitives to supply the values that parameterize the given structure and data type.

Parameter Semantics

TransactionID
This parameter identifies the current transfer session.

FileStructure
This parameter identifies the structure of the file as:

Unstructured, data items in a sequence
Record structured, fixed record length
Record structured, variable record length
Record structured, fixed record length, and indexed
Record structured, variable record length, and indexed

FileDataType
This parameter identifies the type of the selected file as one of:

Text
Binary
Integer
FloatingPoint

FileLength
This parameter estimates the length of the file in octets. This value is passed to the destination host to provide an indication of the storage requirements of the transferred file.

When Generated
The FileParameters primitive may be invoked during the file selection phase provided that the FileParameters option has been negotiated with the participating hosts.

Effect of Receipt
The FTS prepares for transmission of the file using the transfer syntax appropriate to a file of the given structure and data type.

RecordFile

TransactionID
RecordSize (optional)
MaxRecordSize (optional)

Function

The RecordFile primitive is invoked by the user to instruct the FTS to transmit a file as a record structured file.

Parameter Semantics

TransactionID
This parameter identifies the current transfer session.

RecordSize
If the file is composed of fixed-length records, this parameter indicates the record length.

MaxRecordSize
If the file is composed of variable-length records, this parameter indicates the maximum record length.

When Generated

The RecordFile primitive may be invoked during the file selection phase provided that the Record option has been negotiated with the participating hosts.

Effect of Receipt

The FTS prepares for transmission of the file as a file of Record type with the given record length.

IndexedFile
TransactionID
KeyLength

Function
The IndexedFile primitive is invoked by the user to instruct the FTS to transmit a file as an indexed file with the specified key length.

Parameter Semantics

TransactionID
This parameter identifies the current transfer session.

KeyLength
This parameter specifies the bit length of the keys used to index the file.

When Generated
The IndexedFile primitive may be generated during the file selection phase provided that the Indexed option has been negotiated with the participating hosts.

Effect of Receipt
The FTS will prepare for transmission of the file as a file of Indexed type with the given key length.

TextFile

TransactionID
FormatClass
FormatEffectors
CharacterCode
Parity

Function

The TextFile primitive is invoked by the user to instruct the FTS to transmit a file as a file of Text type.

Parameter Semantics (See Section 2.2.2 for a complete discussion of text file parameters.)

TransactionID
This parameter identifies the current transfer session.

FormatClass
This parameter identifies the class of formatting conventions to be used in the transfer.

FormatEffectors
This parameter lists the format effector options to be used in the transfer.

CharacterCode
This parameter identifies the system of encoding text to be used in the transfer.

Parity
This parameter identifies the parity convention to be used in the transfer.

When Generated

The TextFile primitive may be generated during the file selection phase provided that the Text option has been negotiated with the participating hosts.

Effect of Receipt

The FTS will prepare for transmission of the file as a file of Text type with the given parameters.

TabStops
TransactionID
HorizontalTabStops
VerticalTabStops

Function
The TabStops primitive is invoked by the user to inform the FTS of the tab stop locations in a text file that is to be transferred.

Parameter Semantics

TransactionID
This parameter identifies the current transfer session.

HorizontalTabStops
This optional parameter may be repeated multiple times to give the locations of successive horizontal tab stops.

VerticalTabStops
This optional parameter may be repeated multiple times to give the locations of successive vertical tab stops.

When Generated
The TabStops primitive may be generated during the file selection phase provided that the Text option has been negotiated with the participating hosts.

Effect of Receipt
The FTS will prepare for transmission of the file with proper interpretation of embedded tabs.

BinaryFile
TransactionID
BinaryLength
BinaryPacking

Function
The BinaryFile primitive is invoked by the user to instruct the FTS to transmit a file as a file of Binary type with the given parameters.

Parameter Semantics

TransactionID
This parameter identifies the current transfer session.

BinaryLength
This parameter specifies the length of each word of binary data.

BinaryPacking
This parameter indicates whether the binary words of data are to be packed end-to-end in the data stream, or whether they are to begin on octet boundaries.

When Generated
The BinaryFile primitive may be generated during the file selection phase provided that the Binary option has been negotiated with the participating hosts.

Effect of Receipt
The FTS prepares for transmission of the file as a file of Binary type with the specified parameters.

IntegerFile
TransactionID
IntegerLength
IntegerPacking

Function
The IntegerFile primitive is invoked by the user to instruct the FTS to transmit a file as a file of Integer type. All integers will be represented in twos complement notation.

Parameter Semantics

TransactionID
This parameter identifies the current transfer session.

IntegerLength
This parameter specifies the length of each word of integer data.

IntegerPacking
This parameter indicates whether the integer words of data are to be packed end-to-end in the data stream or whether they are to begin on octet boundaries in the data stream.

When Generated
The IntegerFile primitive may be generated during the file selection phase provided that the Integer option has been negotiated with the participating hosts.

Effect of Receipt
The FTS prepares for transmission of the file as a file of Integer type with the given parameters.

FloatingPointFile
TransactionID
FloatingPointLength
FloatingPointPacking

Function
The FloatingPointFile primitive is invoked by the user to instruct the FTS to transmit a file as a file of FloatingPoint type. Floating point data is represented according to the proposed IEEE standard [IEEE, 1981].

Parameter Semantics

TransactionID
This parameter identifies the current transfer session.

FloatingPointLength
This parameter specifies the length of each word of floating point data.

FloatingPointPacking
This parameter indicates whether the floating point words of data are to be packed end-to-end in the data stream or whether they are to begin on octet boundaries in the data stream.

When Generated
The FloatingPointFile primitive may be generated during the file selection phase provided that the FloatingPoint option has been negotiated with the participating hosts.

Effect of Receipt
The FTS prepares for transmission of the file as a file of FloatingPoint type with the given parameters.

5. SERVICES REQUIRED FROM THE LOWER LAYERS

The FTP services require certain services provided by the session layer. The following list of the required services is based on the current ISO session service specification [ISO, 1981].

5.1 CONNECTION ESTABLISHMENT

This service establishes a connection to a given foreign host. A foreign host can accept or reject any connection attempt, and the service places no architectural restriction on the number of connections that may be concurrently established. Connections are established according to quality of service parameters including transit delay, setup delay, and error rate.

5.2 DATA TRANSFER

This service provides for the transparent transmission of data as a stream of octets. Thus, it does not restrict the content, format, or encoding of commands or data. The transmission is sequential and without loss or duplication of data unless explicitly reported.

5.3 TWO-WAY SIMULTANEOUS INTERACTION

This service allows both ends of a connection to concurrently send and receive data.

5.4 ORDERLY RELEASE

This service allows either service user to request the termination of the connection without loss of data. The user invoking this service can determine whether the other user accepts or refuses the termination. If it is refused, the connection is not terminated, and no data is lost.

5.5 P-ABORT SERVICE

This service allows the session layer to terminate the connection for reasons internal to the session layer, when the underlying transport service is unavailable, or when the quality of service falls below specified levels. Use of this service may cause loss of data.

5.6 SYNCHRONIZATION POINT SERVICE

This service allows a user to request a major synchronization point. Such a request completely separates all communication before the synchronization point from communication after it. The session layer services assign a serial number to the synchronization point for possible later use by the service user in requesting the resynchronization service. This service may be used by the FTP to separate and distinguish the phases of FTP operation.

5.7 RESYNCHRONIZATION

This service resets the session connection to a defined state and purges all data not yet delivered. The service user can reset the connection to a previous synchronization point by specifying its serial number.

5.8 EXCEPTION REPORTING

This service notifies the users of unexpected situations not covered by the other services. Typically, these situations are due to service errors or malfunctions.

6. LOWER LAYER SERVICE/INTERFACE SPECIFICATION

TO BE SUPPLIED

7. PROTOCOL ENTITY SPECIFICATION

TO BE SUPPLIED

8. EXECUTION ENVIRONMENT SPECIFICATION

TO BE SUPPLIED

REFERENCES

1. Day, J.D., "A Proposal for a File Access Protocol," ARPANET RFC 520, 1973.

2. EPSS Liaison Group, Study Group 2, "A Basic File Transfer Protocol," HLP/CP (75) 3, Issue 2, INWG General Note 93, 1975.

3. Forsdick, H., and A. McKenzie, "FTP Functional Specification," BDN Report No. 4051, Bolt Beranek and Newman Inc., 1979.

4. Heinze, W., and B. Butscher, "File Transfer in the HMI Computer Network," Proceedings of the 3rd European Network User's Workshop, IIASA, Laxenburg, Austria, p. 13, 1977.

5. IEEE Task P754, "A Proposed Standard for Binary Floating-Point Arithmetic," IEEE Doc. 0018-9162, 1981.

6. ISO, "Data Processing - Open Systems Interconnection - Basic Reference Model," ISO DP 7498, 1981a.

7. ISO, "Draft Basic Connection-oriented Session Service Definition," ISO/TC97/SC16 N700, 1981b.

8. ISO, "The File Service," ISO/TC97/SC16 N578, 1980.

9. Neigus, N.J., "File Transfer Protocol for the ARPA Network," NIC 17759, August 1973. Also in ARPANET Protocol Handbook, 1973.

10. Schicker, P., A. Duenki, and W. Baechi, "Bulk Transfer Facility," EIN/ZHR/75/20, INWG Protocol Note 31, 1975.

11. Simon, G.A., "DCEC Protocols Standardization Program Protocol Specification Report," System Development Corporation TM-7038/204/00, 1981.

12. SYTEK, "DoD Protocol Architectural Model," SYTEK TR-81012, 1981.

APPENDIX A

Extensions

Extensions to the protocol services may consist of new primitives, new parameters, or new classes of protocol service.

A.1. NEW PARAMETERS

The specification of a new parameter for the service primitives has the following format:

Parameter Name: This string provides a mnemonic name for the new parameter.

Primitive Membership: This section lists the primitives in which the new parameter appears and whether each appearance is required or optional.

Purpose: This section describes the purpose of the new parameter.

Semantics: This section describes the general meaning of the parameter.

A.2. NEW PRIMITIVES

The specification of a new service primitive has the following format:

Primitive Name: This string provides a mnemonic name for the new primitive.

Class Membership: This section identifies the class of service to which the new primitive belongs.

Purpose: This section describes the purpose of the new primitive.

Function: This section describes the function of the new primitive and which entity invokes it.

Parameters: This section lists the parameters of the new primitive. The parameter descriptions follow the format given in the New Parameters section above.

When Generated: This section describes the use of the primitive from the point of view of the invoking entity. The description covers such issues as preconditions for invoking the primitive and what is accomplished by its invocation.

Effect of Receipt: This section describes the actions the receiving entity should take on receiving the primitive. This may include immediate actions and actions that may be required at a later time.

A.3. NEW CLASSES

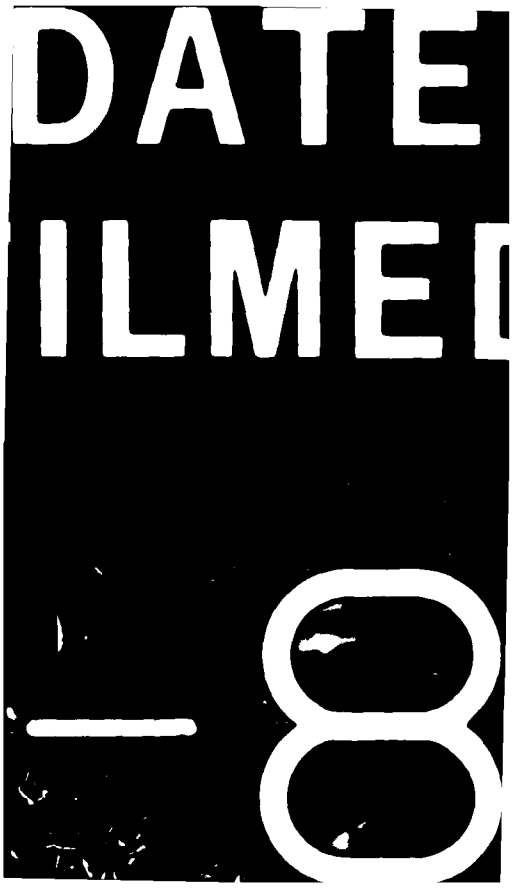
A new class extends the protocol services by providing a new class of service. The specification of a new class has the following format:

Class Name: This string provides a mnemonic name for the new class.

Purpose: This section describes the purpose of the new class.

Service: This section describes in detail the service provided by the new class. It is expected that most new classes will provide services to transfer files with special structure (such as binary or indexed). In this case, this section should describe the structure of these files.

Primitives: This section lists the primitives used to implement the new class. The primitive descriptions follow the format given in the New Primitives section above.


DATE
ILME
8