→ DJ
VSA
3/1/78.

9

A
NETWORK INDEPENDENT
FILE TRANSFER
PROTOCOL

prepared

by the

High Level Protocol Group

A Network Independent File Transfer Protocol

prepared by

The High Level Protocol Group

HLP/CP(78)1

12th December 1977

A Network Independent File Transfer Protocol

prepared by

The High Level Protocol Group

The File Transfer Protocol (FTP) specified in this document is based upon the original 'Basic File Transfer Protocol' defined in HLP/CP (75)3. The facilities and underlying philosophy remain essentially the same; the syntax, however, has been completely re-designed to make it compatible with proposals that are currently being considered for a Job Transfer (or RJE) Protocol. The latter is not discussed here, and protocol elements which are not immediately necessary for implementing the File Transfer Protocol are specified as "undefined" or "reserved", as appropriate. Subsequent issues of this document are likely to elaborate the protocol further, but without requiring compulsory changes to be made to implementations based on the present proposals.

Meanwhile, a parameter is included within the structure of the FTP to facilitate job submission. This is very simple minded in that the full implication is "treat this file as a job, and put it in the job queue". There is no mechanism for setting up an RJE station relationship with the host, and output will not be returned unless some action is taken locally to initiate another file transfer in the reverse direction.

The transfer of a file is a free standing operation, and has no connection with the transfer of any other file, within the definition of the protocol.

The FTP makes no use of the features of a particular network, the only constraints being that errors should be reported and all data delivered in sequence.

Comment and discussion on the contents of this paper should be addressed to one of the working group members in the first instance. Membership of the working group is given in Appendix VII.

Contents

Appendices

Diagrams

1. General Principles

This File Transfer Protocol (FTP) provides a method for the transfer of data, in the form of complete files, across a network, both to and from a remote computer.

The present document defines a protocol designated implementation level B, or FTP-B for short. In designing and specifying the protocol a number of considerations of both immediate and longer term importance have been made. These are set out below:-

The potential differences of size and sophistication between the local and the remote systems have been considered. On the one hand, a relatively simple minicomputer (or even an intelligent terminal) may wish to transfer files to and from a large central computing system with a highly developed filestore, offering a wide range of facilities and controls; on the other hand, a medium to large installation might need to access a small computer providing only the most rudimentary file storage facilities.

The potential for future development is important. It is highly desirable that the protocol should allow all implementations to adapt to changing needs and to develop independently and progressively. In particular, because of the way in which the protocols will be implemented by a very diverse body of users and services, a change at one place must avoid the need for simultaneous changes by others.

The question of timescales is vitally important. The need for a File Transfer Protocol exists now, and so do the operating systems with their respective file storage facilities. Major changes to these operating systems (and in most cases even minor ones) are simply not possible. The protocol must therefore permit the maximum use to be made of existing facilities.

1.1 The Standard File Description

The FTP is built on the premise that a single standardised representation can be set up for a File, or for an organised set of files (a Filestore). This representation can be used in the protocol to express the transactions which are to be performed, mappings being made by each participant to relate the standard descriptions to local resources. The description of this standard conceptual filestore is resolved into a set of distinct characteristics called attributes; the values of these attributes identify or describe the files to be transferred. Some of the attributes of a file have values that are constant during the lifetime of the file (such as Filename); others, however, are important only while the file is being transferred across the network and have values appropriate at that time (such as Mode of Access). The resources mapped onto the standard description, and the mechanism by which the mapping is implemented, will vary considerably from case to case, but the exchange of information will still be in terms of the standard attributes.

Thus, for example:

a) In a computer with a sophisticated filing system based on online disc storage, the mapping will be a matter of changing the formats of the standard attributes to those required by the local operating system. Most of the elements of the standard description will have natural parallels in the filing system interface.

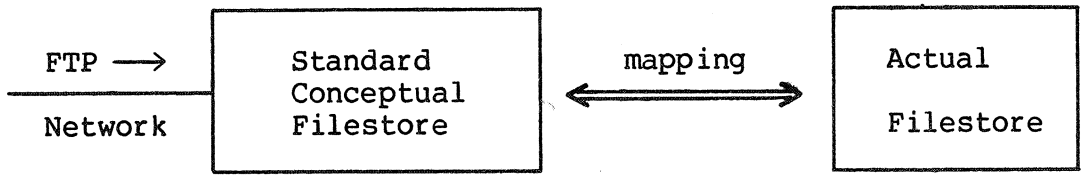


Fig. 1 A Mainframe using the FTP

b) In a small machine, acting as a remote job entry station, the only real resources present might be a card reader and a line printer. In this case, the mapping would be more restrictive than in (a), some of the attributes being constrained to take fixed values, or not having any meaningful equivalents at all. The required transfer, however, can still be discussed within the same framework, the structure of the protocol exchanges being such that declared ignorance of a requested attribute is not necessarily disastrous.

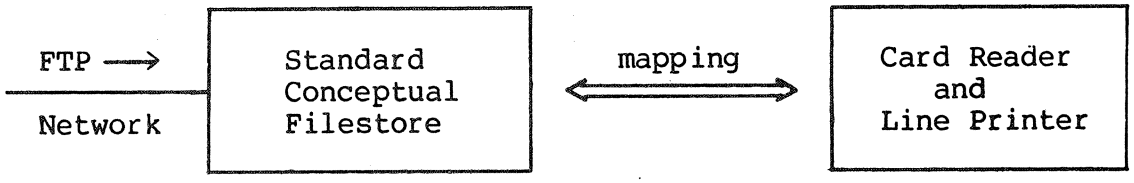


Fig. 2 A Remote Station using the FTP

c) A Host, offering job processing facilities, can draw sufficient parallels between its facilities and such attributes as Mode of Access and Output Device Type to make its services available to users of the FTP. (It is not, however, a part of this protocol to relate input to output for a job, or to provide any information on jobs waiting in the Host.)

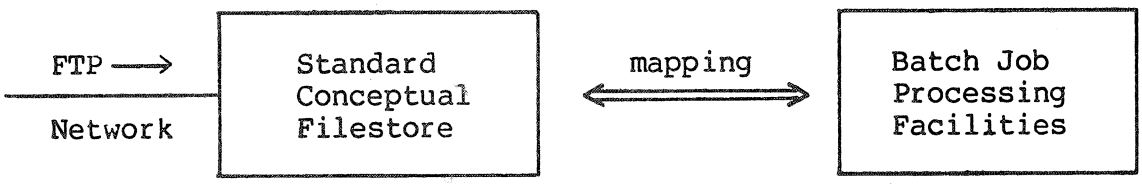


Fig. 3 A Job Server using the FTP

d) One could perhaps envisage a freestanding magnetic tape station backed by a manually administered library. In such a system, the attributes would be mapped into messages for an operator, or derived from his replies. Such a system need only differ from the online filestore in (a), when seen by a remote user, in that it has a longer response time.

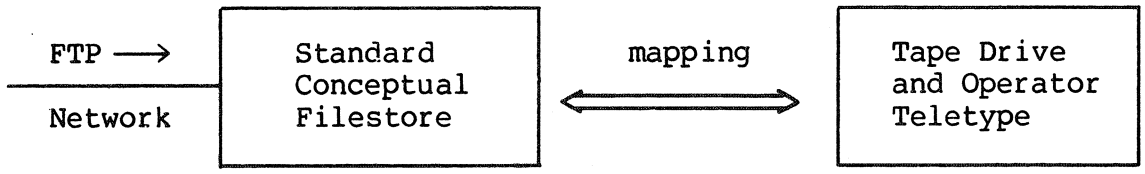


Fig. 4 A Tape Library using the FTP

In general, therefore, the protocol can be used for communication in standard terms with almost any conceivable source or sink of data, whose true nature is absorbed into a mapping local to itself.

1.2 Initiative and the location of attributes

The start of the dialogue which results in the transfer of a file is itself the result of some stimulus outside the scope of the protocol. This stimulus identifies the initiating party, hereafter labelled process P, which begins the dialogue. The other process, labelled Q, is the responder. Process P has some aim, of which process Q is ignorant, in requesting the transfer and so P has the responsibility of guiding the transfer and deciding whether or not it should continue. Q's role in the exchange is solely that of a provider of information, taking only those decisions P requests it to take. Thus, although P has local resources, and must itself relate its local circumstances to the description of the standard conceptual filestore, all exchanges in the protocol are operating on or reporting values of attributes held at Q. P needs the information about Q to direct the transfer, but Q, in its passive role, has no interest in affairs at P.

Figures 5 and 6 illustrate the relationship between process P, process Q and the conceptual filestore.

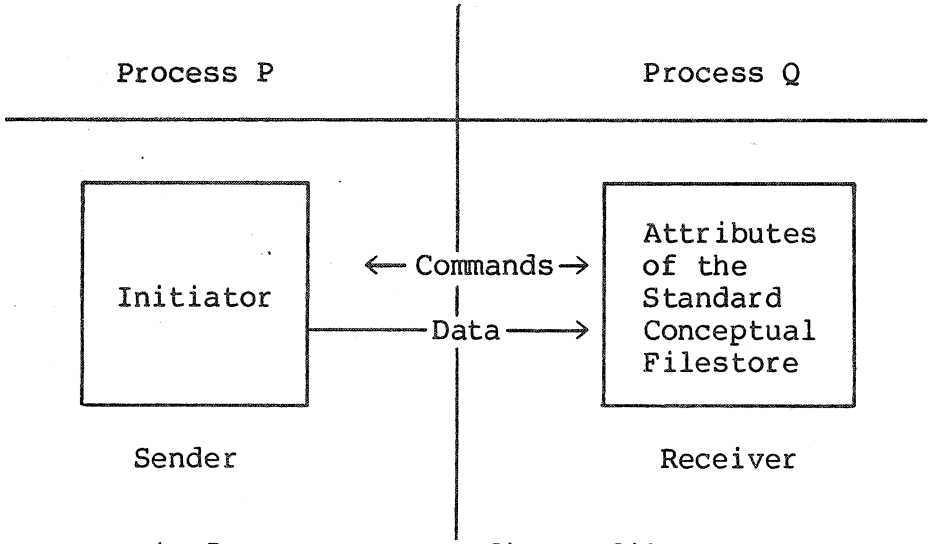


Fig 5. Process P sending a file to process Q

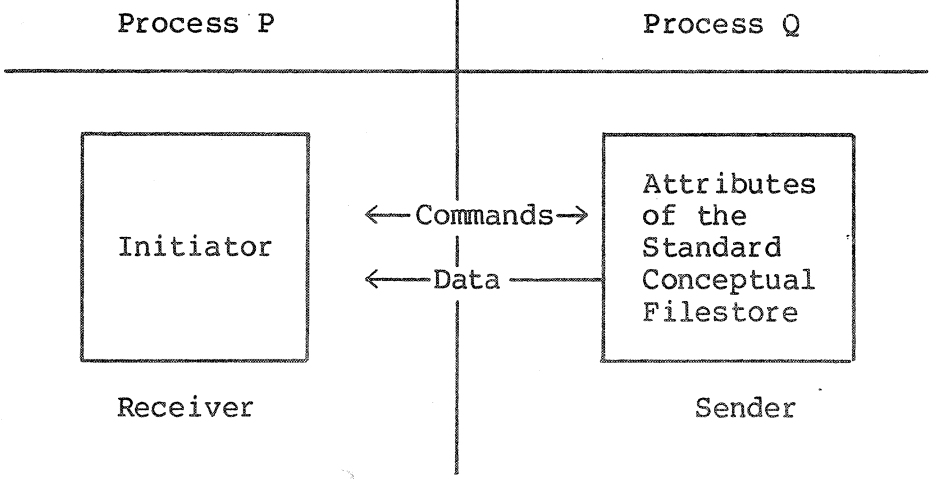


Fig 6. Process P receiving a file from process Q

1.3 FTP general structure

The file transfer takes place in a number of stages. The initial exchanges take the form of commands, each with a number of optional parameters. Each parameter specifies one of the Attributes by using a code number, and gives a suitable value for that Attribute. Once the file has been correctly identified, and the conditions for the data transfer established and agreed, the file data can be transferred. Firm control of the flow of data is achieved by using a second set of commands. The exact details of the different levels of information exchange are given in Section 2 onwards.

One of the most important of the transient Attribute values to be considered during the actual file transfer is

that for the 'Mode of Access'. This can take one of several values which fall into two general groups:-

a) the transfer of data into the filestore, and
b) the transfer of data out of the filestore.

A Start File Transfer (SFT) command specifying a transfer of data into the filestore (such as Make or Append) is termed a 'take' (i.e. the filestore takes the data). An SFT command specifying a transfer of data out of the filestore (such as Read) is termed a 'give' (i.e. the filestore gives the data). This convenient terminology will be used in the following sections.

The diagram below (figure 7) illustrates the general exchange of commands and data in the FTP. In this example a file is transferred from Q's filestore across the network to the process P:

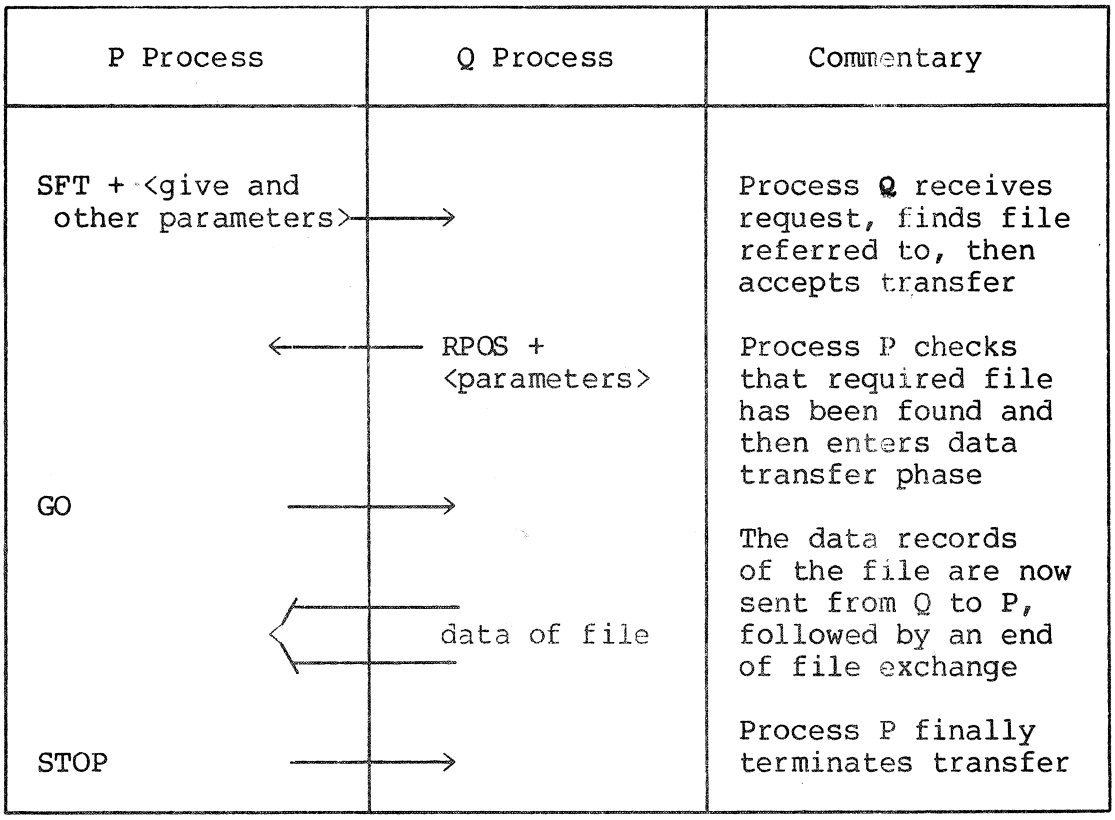


Fig. 7 - The General Mechanism of the FTP.

A 'take', where the file is transferred into Q's filestore, is identical except that the data flows in the opposite direction, from P to Q.

A number of more detailed examples are given in Section 7.

1.4 Transport Service Requirements

Because this protocol is designed to be used on a wide range of different networks, this document does not define a detailed implementation. Instead, it is assumed that each network supports a transport service providing a standard (network independent) set of facilities.

In order to avoid duplication of work at different protocol levels, certain necessary functions are not provided explicitly by the FTP. It is assumed that these functions are provided by the transport service.

In general terms these requirements are as follows:

a) The FTP's basic requirement is for the transmission of a stream of bytes from P to Q, and for the transmission of an independent stream from Q to P. Each stream must be synchronised at the two ends, so that it can be decoded correctly into a sequence of records, and the pair of streams must also be synchronised. The transport service ensures that a well-defined starting point for each stream is provided, by sending a synchronising message around the entire communication loop formed by the two paths. The transport service must regulate the flow of data in each stream so as to limit the demands on the receiver.

b) The transport service is responsible for error detection, and for network dependent error recovery. Ideally, the FTP would prefer to avoid all problems of errors, because the transport service should be 'error free'. More realistically, it is required that undetected errors should be sufficiently rare (according to some user-chosen 'grade of service') and that errors detected but not corrected by the transport service should be reported to both the File Transfer processes in a manner which permits recovery of synchronisation. Such an event is called a transport service reset (see Section 2.7). If a Reset cannot be given, both processes must be warned that the communication has broken down irrecoverably.

c) The transport service must provide a function to force the delivery of any currently buffered data, and this function should be used by the FTP after any command has been issued, to prevent deadlocks. (The transport service would otherwise buffer data in order to use communication facilities efficiently.)

d) A process may also issue a synchronise command (Reset), which has the same effect as a transport service detected error. This command is not essential to the operation of the protocol, but it may be used to simplify the restart procedures discussed in Section 2.4 and in Appendix V.

e) In error situations a process may, as a last resort, close the transport service connection in order to terminate the transfer unilaterally. Catastrophic failure of the transport service may also result in a spontaneous close.

2. Protocol Structure

The protocol consists of three phases: initialisation, data transfer, and termination. The initialisation phase establishes the identity and properties of the file and the details of the format in which it is to be transferred. The data transfer phase includes the actual data transfer and the administrative exchanges required to regulate it. In this phase, the receiver can:

a) abort the transfer completely;
b) stop the data flow temporarily;
c) request re-transmission of data.

The termination phase ends the dialogue and indicates the final state of the transfer.

The details of the Level Ø and Level 1 commands referred to in this section will be found in Sections 4 and 6 respectively, with illustrated examples in Section 7. (Detailed knowledge of these commands is not required on a first reading of this section.)

2.1 Levels of Information Exchange

Within the FTP there are three levels of information exchange:

Level Ø - Process initiation and control;
Level 1 - Data transfer control;
Level 2 - Data.

Level Ø is in force when the file transfer is initiated, and basic characteristics of the transfer (such as mode of access, allowed codes, use of compression, etc.) are agreed by an exchange of Level Ø commands. The initialisation process also identifies the file transferred, and may convey information about that file. All the information used in the initialisation is passed by parameters of the Level Ø commands, the values affecting or reflecting the values of the attributes at Q.

Level 1 commands allow error reporting, restarting, pausing, acknowledgement and change of data encoding. They occur in the data transfer phase, which begins after the Level Ø GO command has been issued by process P.

The data transfer phase must begin with a Start of Data (SS) command. Thereafter, the information flow consists of Level 2 data going from sender to receiver, possibly with some Level 1 commands embedded. Level 1 commands may also flow in the opposite direction. Anything which is not a Level 1 command belongs to Level 2 until the receiver has confirmed 'End of Data' by means of the

appropriate Level 1 command (ER(OK) or ER(E)). This command terminates the data transfer phase and causes Level Ø to be re-entered.

The file transfer as a whole is terminated when process P sends a Level Ø STOP command, which can include textual diagnostic information. The provision of additional diagnostic information is for further study.

2.2 Initialisation Phase

The initialisation phase identifies the file to be transferred, establishes its properties and at the same time sets up the values of the various transfer attributes. This is accomplished by the exchange of two commands. Firstly, P states its requirements in the parameters of the SFT command. Q then considers these values in the light of its knowledge and constraints, and returns a reply which may be either accept (RPOS), or reject (RNEG); P only continues to the data transfer phase if a positive reply is received. P checks the received values and either starts the transfer (with a GO command) or terminates the exchange. Examples of possible exchanges can be found in Section 7.

The process by which the target values are found is one of selection; each parameter which is marked by its qualifier (see Section 3.4) as a selector gives some constraint reducing the set of files eligible for transfer. Thus one may select a file with Filename equal to a given value, or any file for a given Username, or any file of less than a given size. All the selectors given amongst the parameters of the SFT command combine to restrict the set of possible files for transfer. The transfer can only proceed if some file is found which satisfies all the constraints; if the selection permits more than one file, then the choice of a particular file rests with the local implementation.

Parameters which are not selectors are modifiers. Once a file has been selected as above, any modifiers present change the description of the file at Q. Attributes can be set precisely, by making them equal to the given value, or constrained more weakly by requiring that they be, for instance, less than or equal the given value. The use made of these modifiers in the FTP is not as large as that expected in future File Interrogation and Job Transfer Protocols.

A final important aspect of the initialisation phase is the provision of information for P in the reply commands. To request this, P can set a monitor flag in the qualifier of any SFT parameter. Q is then required to include a statement of the value of that attribute with its reply. A value can be monitored, without any other effects, by

placing the monitor flag in a dummy selector which places no constraints on the value.

Evolution of the Protocol and variation in the sophistication of applications may result in Q being faced with a parameter referring to an attribute of which it has no knowledge (simple implementations may not include all the currently defined attributes). In this case, all selection and modifying operations are to be considered successful, and the reply 'unknown attribute' given to any monitor request. A standard reply is also available for use where the attribute is recognised but a value is not available.

Implementations permitting the use of advanced facilities should be prepared for the refusal of these facilities, and be able to operate at the simplest level (see, for example, the Facilities attribute (number [ØE], Section 5.12)).

2.3 Data Format and Structure

The FTP provides transparent transfer of a stream of bytes; the protocol does not necessarily constrain how this stream is to be interpreted, and special interpretations can be agreed between the two parties, outside the protocol. Any interpretation applies to the byte stream as transmitted; it does not force the file to be stored or retrieved in any particular form.

It is recognised, however, that there are certain common applications, such as transfer of text for a printing device; attributes are therefore provided to describe the intended implementation in these cases, but the values of these attributes do not change the actions of the protocol in any way. There are two major classes of interpretation - text and binary. In either case the data is divided into records whose significance is determined by the user.

The Codes attribute (number [Ø2], Section 5.3) specifies the set of codes, including binary, which may be used during the transfer; the Level 1 Code Select (CS) command is used during the transfer to specify whether the following records are text in a particular code, or are binary.

The intended layout of the text on the printed page may be specified in several different ways; the Format Effectors attribute (number [Ø3], Section 5.4) identifies the technique to be used. By default, each record corresponds to a single printed line, successive lines being separated as if by carriage return and line feed. Alternative techniques are to use the first character of each record as a carriage control (as in Fortran), or to ignore the record structure and to specify formatting by embedded control characters.

In binary sequences, introduced by the Level 1 Code Select (CS) command, the bytes are considered as a packed sequence of binary words. The word length and packing technique are specified by the Binary Mapping attribute (number [Ø4], Section 5.5).

2.4 Restart Facilities

The FTP provides for recovery from data loss, by re-transmission of part of a file. If this form of recovery is to be used then the value of the Facilities attribute (number [ØE], Section 5.12) must allow Restart Requests and Restart Mark Acknowledgement (value [ØØØC]). The sender provides this facility by placing restart marks at convenient points in the transmitted data. The receiver acknowledges when data has been accepted up to a given mark, and may request re-transmission of unacknowledged data from a mark.

For example:

a) When output is to a line-printer, it is desirable to guard against loss of data caused by a paper jam or other mechanical failure. This would require the receiver to buffer several pages of text, an impracticable amount on a terminal without backing store. The sender, however, has the entire file stored in its filestore, and can easily retransmit the required data. If the sender places a restart mark at the start of every page, then a local operator command to the receiver, such as 'backspace 5 pages', could be mapped into a request for re-transmission from the last mark less 5.

b) If data is to be sent from a volatile data source (such as a card reader, or a gateway to another network) into a filestore, the roles are reversed. The receiver now has primary responsibility for the protection of the data, and must send acknowledgements to the sender to indicate when it can throw away buffered copies of the input data. The mark spacing must be small, since the sender will only be prepared to buffer a very limited amount of data.

c) The mechanism can also be used to recover from network failures reported by the transport service during the data transfer phase. A failure will result in resynchronisation, leaving both of the communication paths empty. The receiver knows the last restart mark which it received, but knows neither the last mark sent nor the last mark whose acknowledgement has been received. It must, therefore, repeat its last acknowledgement (MR) and send a restart request (RR) command to continue from the last mark received.

Although each of these examples places different requirements on the restart mechanism, the underlying commands and mechanisms are the same. The receiver sends a Restart Request (RR) command, quoting the last mark preceding the error, and then searches its input until the matching Start of Data (SS) command is received. The sender, on receiving an RR command, abandons the current data output, sends an SS command quoting the same mark as the RR command, then re-transmits the data following that mark. (There is an example of the use of these mechanisms in Section 7.2.4.)

The Acknowledgement Window attribute (number [ØA], Section 5.9) puts a limit on the number of marks which can be sent without being acknowledged by the receiver, and therefore on the volume of data which the sender needs to keep in buffers. The sender should not transmit data after sending the last permitted mark, until some further acknowledgement has been received.

Restart marks are placed in the data stream at points chosen by the sender, but the positioning of such marks must be repeatable. The marks may be acknowledged by the receiver either singly or in groups, in that acknowledgement of a particular mark implies acknowledgement of all previous marks.

2.5 Continuing in a new transfer

The restart facilities described above provide for recovery from limited losses of data within a single file transfer operation. In some circumstances, however, if there is a substantial break in the network, the file transfer may be completely interrupted. Alternatively, it may be operationally desirable to interrupt a file transfer, for example when a machine has a scheduled break in service. These circumstances require a different form of recovery, since information must be retained between two (or more) separate file transfer operations. The subsequent file transfers can use the Initial Restart Mark attribute (number [ØB], Section 5.1Ø) to specify the information to be transmitted. It may also be necessary to indicate that the subsequent transfer is a continuation of the first, although details are not included in this protocol.

In this case, the recovery mechanisms are:

a) Recovery initiated by the receiver.

   If the transfer is aborted completely, and is to be resumed by the receiver, the procedures are fairly simple. No acknowledgements need be used; the receiver merely keeps a count of marks received, starting from the Initial Restart Mark value. When

resuming the transfer, the receiver quotes the last mark received as the Initial Restart Mark, and sets all other attributes equal to their values for the previous transfer.

b) Recovery initiated by the sender.

In many practical cases, the receiver is a relatively simple system, and is unwilling or unable to store the restart information. The sender may, in a particular implementation, already have a record of the transfer to be made, such as the relevant entry in a job output queue, and only needs to add the Initial Restart Mark to the information already stored. Mark Acknowledgement must be used, and the sender records the last mark acknowledged for use in the restart attempt. (It must, of course, record the full mark number, not merely the least significant 8 bits appearing in the MR command.)

2.6 Pauses in the data flow

There may be circumstances in which the receiver discovers that it has conflicting demands for resources, which would be best dealt with by a high level scheduling decision to suspend a transfer, rather than relying on low-level flow control. In such circumstances, the receiver can request the sender to hold by using the hold form of the Quit (QR) command. The sender issues a temporary End of Data as soon as convenient; the sender is then in the hold state. While in the Hold state no data may be transmitted; Hold state is cleared by the hold form of the end acknowledge (ER(H)) command, or by certain error conditions.

If some data has been lost or discarded, the receiver can issue a Restart Request command before cancelling the Hold state. The sender would then recommence with an SS command when the releasing ER(H) was received. There is an example of the use of temporary Hold in Section 7.2.4.

2.7 Transport Service Reset

Following a transport service reset (see Section 1.4b), the two parties are synchronised and there is no data in transit. However, some previously transmitted information may have been lost, so that there may not be agreement between the two processes as to the state of the transfer. In order to regain agreement as to the state, certain actions are prescribed.

a) If the transfer is at Level Ø, the transport service connection should be closed, since the transfer either has not yet started, or has already finished.

b) In the data transfer phase, a transport service reset clears any Hold state, and causes the truncation of any incomplete control handshake. Before the transfer of data can continue, the state must be regularised, by the receiver sending a Mark Acknowledge (MR) command, usually repeating the last mark acknowledged, and a Restart Request (RR) command indicating the point at which data transfer is to be restarted. Alternatively, if a temporary Hold had been established, there is no ambiguity and the receiver can send the hold form of the Quit (QR) command to re-establish the Hold state. If continuation is not possible, the transfer may be terminated indicating an error (by the receiver sending a QR(E) command, or the sender an ES(E)).

2.8 Time-outs

Loss of communication or process failure could occur at any time, and automatic recovery may not be feasible or successful. To help with such situations the protocol includes (optionally) the notion of time-outs. The time period used is held as a transfer control attribute, Minimum Time-out Interval (number [ØD], Section 5.11).

This value can be monitored or adjusted by process P, and is used to time all situations where a response, interaction, or continuation is required. The result of a time-out maturing is invariably 'catastrophic' to the transfer, which will be terminated unilaterally by the timed-out process. The Transport Service may provide non-destructive time-outs for error recovery. The default value of the Minimum Time-out Interval attribute, 1Ø minutes, has been chosen to allow for operaror action, such as mounting a disc pack or reloading a printer.

The action to be taken on time-outs is suggested in each of the following sections where appropriate.

3. Command and Data Formats

3.1 Terminology and Representation

In the sections which follow, various conventions are used to represent values or formats:

a) Numbers appearing in square brackets are hexadecimal representations of values, a pair of digits standing for an 8-bit field and four digits for a 16-bit field. The most significant byte of a 16-bit field is transmitted first.

b) Bits are numbered from the least significant (right-hand) end of a field, the least significant bit being numbered zero. Normally both the bit number and the corresponding hexadecimal representation for that bit are given, thus:

bit 5 or [20]

c) Alphanumeric values, appearing in double quotes, represent the given IA5 string, with any parity, prefixed by an 8-bit byte giving its length. The length of the IA5 string does not include the length byte itself.

"LP" is equivalent to the three bytes: [02],'L','P'
"" is equivalent to the single byte: [00]

d) Non-specific values occur as follows:

n	- a hexadecimal digit
nn	- an 8-bit field
nnnn	- a 16-bit field, used subdivided
N	- a 16-bit field, used as a single integer
string	- an alpha string, prefixed with a length byte, as in (c).

e) Information levels are as defined in Section 2.1, that is:

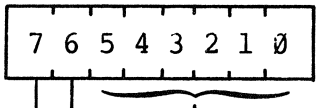
Level 0 - Process initiation and control;
Level 1 - Data transfer control;
Level 2 - Data.

f) The abbreviation 'k' is used for multiples of 1024 ([0400]).

3.2 Basic Record Format

All information transmitted by the FTP is divided into records. Data and Level Ø commands are carried in records of arbitrary length; for transmission a record is divided into sub-records, each of which is preceded by a header byte. This structure both permits records of arbitrary length and permits compression of repeated bytes. The sub-record boundaries permit the insertion of Level 1 commands. These commands have a fixed length (two bytes), and are preceded by a special zero header byte.

The structure of the header byte is:



(End of record) EOR
If set, this bit indicates that the following data is the last sub record of the current record.

(Compressed sequence) COMP
If set, indicates that a single 8 bit byte follows which must be inserted CNT times.

(Count) CNT
A binary count (max 63)

The meaning of the count is dependent on the setting of the compression bit COMP (bit 6 or [4Ø]). If bit 6 is not set then the count gives the number of 8-bit bytes which form the sub-record. The maximum size of a sub-record is 63 bytes. The count does not include the header byte. If bit 6 is set, then the count gives the number of times that the single byte following the header must be inserted when reconstructing the compressed sub-record.

The header of a null record will always have EOR (bit 7 or [8Ø]) set and so the byte will be non-zero. A sub-record of zero length with EOR not set ([ØØ]) is expressly reserved. This enables a zero byte to be used as a special header for introducing Level 1 commands, which are always of fixed length (two bytes) and not compressed. This mechanism allows Level 1 commands to be recognised, in principle, at any time.

There are no implicit restrictions concerning the use of compressed sub-records at Level 2. These may be used wherever needed (given that the receiving process has previously agreed to their use).

Section 7.3 gives some examples of the use of this record structure.

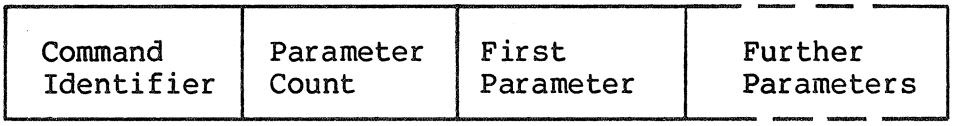
There is no explicit facility for transferring data records of fixed length.

3.3 Command Formats

In Level Ø and Level 1 of the protocol, information is transferred in the form of commands. Level 2 is the body of the file, and there are no restrictions on its content.

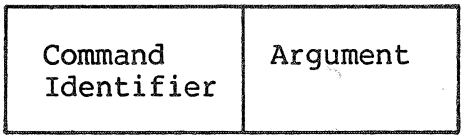
A command is made up of the following components, transmitted in a single record in the order shown:-

Level Ø



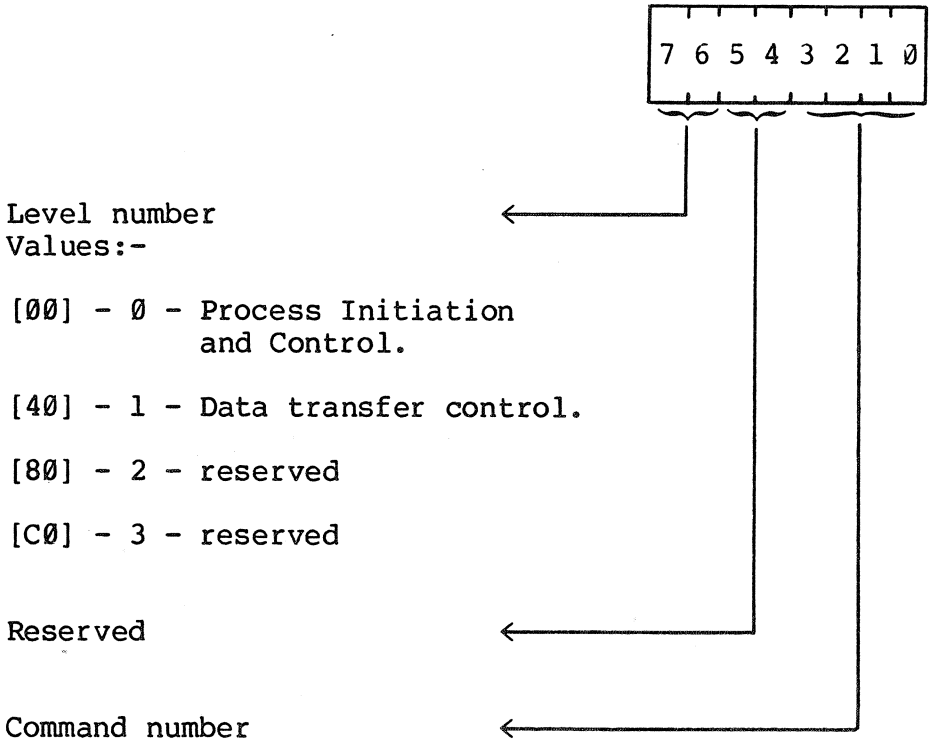
Level Ø commands may not be compressed (even after compression has been agreed by both ends); but they may if necessary be split up into several sub-records.

Level 1



3.3.1 Command Identifier

The command identifier is a single 8-bit byte giving the information level and the command number, thus:-



The command numbers are given in Sections 4 and 6.

3.3.2 Parameter Count or Argument

This is a single 8-bit binary integer.

For Level Ø commands it is a count of the number of parameters given with the command.

For Level 1 commands it is an argument with a variety of values or settings, depending on the command in use. Level 1 commands never have parameters.

3.4 Parameter Format

Parameters are only used with Level Ø commands. There may be as many parameters as are necessary, in any order; the number present is given by the parameter count. They refer to attributes of the file (or filestore), or to attributes of the data transfer itself, considered to reside at process Q. The attributes and their particular functions are defined in Section 5 and Appendix III.

Each parameter is made up of three parts:-

a) Attribute Number
This is the first 8-bit byte of the parameter.

b) Qualifier
This establishes the specific function of the parameter. It is transmitted as the second 8-bit byte of the parameter. It is defined in detail below (Section 3.4.1).

c) Value
Most, but not all, parameters require a value to augment the qualifier. The form of this value is indicated by the format field of the qualifier, and the purpose by the type of parameter bit of the qualifier.

In commands issued by the P process, each parameter has one of two purposes:-

i) To change the current value of the attribute according to a constraint given by the parameter - modify.

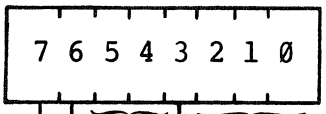
ii) To be used, in conjunction with the other parameters of the command, to select the file which is to be accessed - select.

In this case a value supplied with the parameter is used only for comparison with existing values (such as the Filename).

In commands issued by the Q process, each parameter describes the current value of the corresponding attribute; the format of the value must be present in the qualifier, the operator should be Equal, and the monitor bit should be set to zero. (See Section 2.2.)

3.4.1 Parameter Qualifier

The qualifier has been designed with potential for expansion of its functions in the future. Some of the settings are only meaningful, at present, when used by process P, and should be avoided in qualifiers used by Q.



MONITOR (for P only)
[00] - Do not include in reply command
[80] - Include amongst parameters of reply command.

Reserved for future use; should be set to zero

FORMAT OF VALUE
Values:-

[00] = 0 - Null (value absent); the attribute is unknown

[10] = 1 - Null (value absent); no value available for this attribute

[20] = 2 - Number (as a 16 bit binary integer)

[30] = 3 - Alpha (IA5 string; see Section 3.1c)

TYPE OF PARAMETER

[08] - Modify; change current value under the constraint given by this parameter.

[00] - Select; use given value to control the selection of an entry (e.g. Filename).

OPERATOR

The operator and value together define the constraint under which the value of the attribute is to be modified, or an item is to be selected. Q should only use the Equal operator. The defined encodings of the operator field are given overleaf.

This table gives the representation of the possible selection operators. The selection is successful if

attribute value <operator> parameter value

is true.

[Ø2] - EQ Equal
[Ø3] - LE Less than or Equal
[Ø5] - NE Not Equal
[Ø6] - GE Greater than or Equal
[Ø7] - ANY Any value acceptable

Other values are reserved for future use.

For strings, only the operators Equal, Not Equal and Any are defined.

For numbers, the operators have their usual arithmetic meanings.

For fields of bits, one value is Less than or Equal another if it is exactly equal, or can be obtained by replacing some one bits by zeros. One value is Greater than or Equal to another if it is exactly equal, or can be obtained by replacing some zero bits by ones.

Common values found for the qualifier are:

a) [2A] Modify the attribute value to be equal to the given value, which is a 16-bit integer.

b) [32] Select an entry with the value of the given attribute equal to this value, which is an alphanumeric string.

c) [3A] Modify the attribute value to be equal to the given value, which is an alphanumeric string.

d) [AB] Modify the attribute value so that it is less than or equal to the given value, which is a 16-bit integer. Return the value of the attribute in the reply command.

e) [AE] Modify the attribute value so that it is greater than or equal to the given value, which is a 16-bit integer. Return the value of the attribute in the reply command.

3.4.2 Parameter Value

The format of the parameter value is specified by the corresponding qualifier. The interpretation depends on the particular Attribute, and may be:

a) Null, in which case no value is present.

b) Alphanumeric, being a length byte followed by a string of that length (not including the length byte). The string consists of IA5 characters of any parity.

c) Numeric, being a 16-bit integer, the most significant byte being transmitted first.

d) Field of bits, being two bytes, used as 16 independent indicators.

4. Process Initiation and Control - Level Ø Commands

The commands used to control and define the file transfer are as follows:-

Command Number	Name	Use	Used by
Ø	STOP	End of Transfer	P
1	GO	Start Data Transfer	P
2	RPOS	Reply - positive	Q
3	RNEG	Reply - negative	Q
4	SFT	Start File Transfer	P
5-15		Reserved	-

Each command, with its optional parameters, is described in the following sections.

The structure defined in Section 1 is referred to throughout:

a) The file transfer takes place between the initiating process P and the responding process Q (either to or from that process).

b) Process P always initiates the file transfer (with SFT) and the data transfer phase (with GO), and eventually terminates the interaction (with STOP).

c) All the parameters quoted, by either participant, affect or reflect the values of the attributes in the standard conceptual filestore at Q, no matter whether the file is being transferred to or from Q.

Level Ø commands can be of any length and are transferred using the record structure described in Section 3. A record may only contain one command. The representations defined in Section 3.1 are also used here.

4.1 Start File Transfer (SFT)

Command: [Ø4] [nn] <optional parameters>

Function:

This command is sent by the initiating process P, accompanied by all the parameters necessary to define the details of the transfer and to identify the file. [nn] is the number of parameters (each with an attribute code number, a qualifier and an optional value) which follow.

4.2 Reply Positive (RPOS)

Command: [Ø2] [nn] <optional parameters>

Function:

This command is sent by process Q to accept the proposed transfer and is accompanied by the parameters defining its view of the details of the transfer of the file. The command should include the values of any attributes for which monitoring was requested, together with the values of any which have been changed. [nn] is the number of parameters which follow.

4.3 Reply Negative (RNEG)

Command: [Ø3] [nn] <optional parameters>

Function:

This command is sent by process Q to indicate that the proposed transfer may not take place. Parameters should be provided giving suggested values for those attributes which contributed to the failure; this information may be used, outside the protocol, to plan a further File Transfer. Additionally, the parameters will normally include at least a value for the State of Transfer attribute (number [ØF], Section 5.13) which gives an encoded reason or other indication of the difficulty (such as named file not found). With a negative reply, Q is not required to act on monitor requests. [nn] is the number of parameters which follow.

4.4 GO

Command: [Ø1] [nn] <optional parameters>

Function:

This command indicates that the P process, after considering the reply, is entering the data transfer phase.

Normally this command has no additional parameters (i.e. [nn] is zero), but an option is provided in the Facilities attribute (number [ØE], Section 5.12) to allow experiments with extended exchanges. Parameters of the GO command might be used to adjust certain transfer attribute values, for instance to allow the possible data codes to be reduced to only one. (Anyone interested in performing such experiments should contact a member of the working group - see Appendix VII.)

4.5 STOP

Command: [ØØ] [nn] <optional parameters>

Function:

This command is sent by the P process to indicate the final termination of a transfer. It may be accompanied by parameters giving the final status of the file after transfer, and possibly messages or other information. If the STOP is being used to reject an RPOS, the parameters should include the values of those of the RPOS which proved unacceptable.

4.6 Time-outs and Level Ø

a) Process P can send STOP after a Time-out and could then attempt a totally new transfer once communication has been re-established. (There is no possibility of time-out after STOP since it is the final termination of the transfer.) Alternatively, if a STOP command cannot be sent, P may close the transport service connection without transmission of any further command.

b) After a time-out at Level Ø, the process Q should close the transport service connection, giving any diagnostic information that the transport service is able to convey.

4.7 Parameters of Level Ø Commands

The possible parameters for use with Level Ø commands are detailed in Section 5. Section 2.2 discusses how these commands are used to specify the actions to be performed.

5. Parameters Describing Standard Attributes

The following sections give details of the possible parameters of the Level Ø commands. Each parameter refers to a particular Attribute and its value held at Process Q, and is named after that attribute. An attribute may be referred to at most once by a selector or modifier in a single command. (The use of more than one parameter referring to the same attribute in a command is a subject for further study, but may well be required in a Job Transfer Protocol.)

The function and purpose of each Attribute and the possible values it can have are given. The optional parameters have three parts:

<Attribute Number><Qualifier><Value>

(In some cases the <Value> part is absent; this is indicated by the qualifier. The definitions of components of an optional parameter are given in Section 3.4.)

A suggested qualifier for use by P is included with the description of each parameter. Q should use a qualifier giving the format of the value and the Equal operator. The use of the Type of Parameter and Monitor bits by Q is for further study; at present they should be set to zero.

The default value given with each parameter description will be taken if neither process uses the parameter. However, the default does not itself place any constraint on the value Q can quote in a reply.

The first group of parameters provide access to Transfer Control Attributes (with numbers [Øn]). These parameters establish the rules under which the transfer will proceed.

The remainder of the parameters deal with Identification and other related Attributes. These parameters allow the correct file to be selected for the transfer.

A list of the Attributes to which the parameters refer, and their defaults is given in Appendix III.

The representations used in this section are those described in Section 3.1.

5.1 Protocol Identification

Parameter: [ØØ] [2A] <nnnn>

Default value: [ØØØØ]

Function:

The protocol identification parameter gives the protocol name, protocol version and implementation level of the File Transfer implementation. The first hexadecimal digit of the value identifies the protocol, being zero for FTP. The second hexadecimal digit gives the protocol version, FTP-B being coded as zero. The remaining two digits are available for the encoding of the local implementation level.

5.2 Mode of Access

Parameter: [Ø1] [2A] <N>

Default value: [ØØØØ]

Function:

The mode of access parameter is used to define the direction and mode of access to be used in the proposed transfer.

Specification:

Bit 15 (or [8ØØØ]) of N is used to indicate the direction of the proposed transfer, thus:-

[ØØØØ] P->Q (TAKE)
[8ØØØ] Q->P (GIVE)

The following values for N are defined:-

[ØØØ1] MAKE ONLY
A file is to be written to the filestore of process Q, creating a new file with the name given by the Filename parameter. RNEG will result if a file already exists with the given name.

[ØØØ2] REPLACE ONLY
A file is to be written to the filestore of process Q, deleting (emptying) the existing file of the quoted name, before writing to it. RNEG will result if no file of that name exists.

[ØØØ3] REPLACE OR MAKE
Combination of modes [ØØØ1] and [ØØØ2] for use when it is not known whether a file exists. The transfer is to proceed notwithstanding.

[ØØØ4] APPEND ONLY
Requests that the file to be transferred to process Q should be appended to the existing file of the quoted name. RNEG results if the file does not exist.

[ØØØ5] APPEND OR MAKE
Combination of modes [ØØØ1] and [ØØØ4] for use when it is not known whether a file exists. The transfer is to proceed notwithstanding.

[2ØØ1] TAKE JOB INPUT
The file to be transferred to process Q should be injected into the job stream of the receiving host. The job name is given by the Filename parameter, if present.

[4001] TAKE JOB OUTPUT
The file being transferred to process Q should be sent to the device specified by the Output Device Type parameter. It may be spooled before final delivery to the device.

[8001] READ AND REMOVE
A file of the quoted name is to be read (copied) from the filestore of process Q, and then deleted. The file will not exist on that filing system after being transferred successfully.

[8002] READ ONLY
A file of the quoted name is to be read (copied) from the filestore of process Q.

[8004] DESTRUCTIVE READ
The file is read (copied) from the filestore of process Q, and deleted progressively as receipt of each part is acknowledged.

[A001] GIVE JOB INPUT
The file to be transferred from process Q is a complete job, to be executed in P's jobmill.

[C001] GIVE JOB OUTPUT
The file to be transferred from process Q will be removed from the queue of output files of that host.

Comments:

1. JOB INPUT (modes [2001] or [A001]) does not set up an RJE station to host relationship, but provides a simple scheme to enable jobs to be put into the job stream for subsequent processing. The actual file transferred should contain all the necessary job control to run the job. Once the file has been received and placed in the job stream, the FTP is not responsible for maintaining a record of the source. Automatic output recovery is not directly a function of the FTP.

2. The default setting of the Mode of Access Attribute is [0000] which is 'No Access'. Thus if this parameter is omitted an RNEG will certainly result.

5.3 Codes

Parameter: [Ø2] [AB] <nnnn>

Default value: [1Ø01]

Function:

This parameter defines the code or codes that may be used in the data transfer phase.

Specification:

Each binary digit of nnnn is used to indicate that the defined data code or parity option can be used. The following bits have been defined:-

Bit	Value	Code
Ø	[ØØØ1]	IA5
1	[ØØØ2]	Binary
2	[ØØØ4]	EBCDIC
3	[ØØØ8]	Private Codes
4-11	[ØFFØ]	reserved
12	[1ØØØ]	Any Parity
13	[2ØØØ]	Odd Parity
14	[4ØØØ]	Even Parity
15	[8ØØØ]	Parity Bit Zero

Comments:

1. The default data code set is IA5 with any parity (i.e. default value = [1Ø01]).

2. Code selection in the data transfer phase is indicated by the use of the Level 1 CS command (see Section 6.3). After an SS command, the selection is IA5 with any parity.

3. The Q process can adjust the choice of codes and report the change by using the RPOS command. With the suggested qualifier, the set of codes and parity options can only be reduced, and not increased, if the transfer is to proceed.

4. The parity option is ignored when EBCDIC or binary is selected in the data transfer phase.

5. IA5 will be used for alphanumeric values in the initialisation phase, even if not permitted in the data transfer phase.

6. The recommended qualifier leaves Q the possibility of reducing the code set in the RPOS command, throwing the responsibility for conversion onto P. If Q is the sender, this reduction enables P to free translation resources, but is not otherwise necessary, since Q can simply refrain from using the particular code.

5.4 Format Effectors

Parameter: [Ø3] [AB] <nnnn>

Default value: [ØØØ1]

Function:

The parameter gives the formatting implications of the record structure in the data stream, the set of data characters with formatting significance and the actions to be associated with those characters.

Specification:

The defined bits of nnnn fall into three groups:

a) Actions associated with the record structure of the Level 2 data.

Bit	Value	
Ø	[ØØØ1]	The end of each data record corresponds to the start of a new line when formatting the data.
1	[ØØØ2]	The first character of each record is an ANSI (Fortran) carriage control character, giving the formatting action to be taken before processing the record.

If neither of these bits is present, the division of the file into records has no formatting significance.

b) Characters which have formatting significance when found within the data.

Bit	Value	
4	[ØØ1Ø]	CR Carriage Return
5	[ØØ2Ø]	LF Line Feed
6	[ØØ4Ø]	NL Newline
7	[ØØ8Ø]	FF Form Feed
8	[Ø1ØØ]	VT Vertical Tab
9	[Ø2ØØ]	HT Horizontal Tab
1Ø	[Ø4ØØ]	BS Backspace

c) Characters which are to have additional side effects.

Bit	Value	
12	[1ØØØ]	CR implies LF action.
13	[2ØØØ]	LF implies CR action.

Comments:

1. The default setting, [ØØØ1], if this parameter has not been used, is 'treat each record as a single (unformatted) print line'.

2. The use of this parameter has no effect on binary sequences in the data stream.

3. The characters above are referred to by name, not by their value in any particular code; for example, HT is [Ø9] in IA5, but [Ø5] in EBCDIC.

4. Bit 12, [1ØØØ], is only meaningful if bit 4, [ØØ1Ø], is set; bit 13, [2ØØØ], is only meaningful if bit 5, [ØØ2Ø], is set.

5. The effect of control characters within the data, other than those listed, is not defined by the protocol; if a control character is not selected as having formatting significance, the effect of its use is not defined by the protocol.

6. The suggested qualifier gives Q the option of reducing the set of available formatting functions. P must either perform any necesary conversions or abort the transfer.

5.5 Binary Mapping

Parameter: [Ø4] [2A] <nnnn>

Default value: [ØØØ8]

Function:

This parameter is used to define the mapping of binary data onto the 8-bit bytes of the data stream, when the binary code is selected.

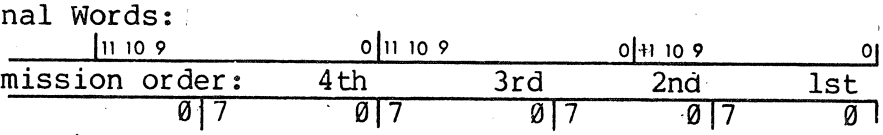
Specification:

The first pair of digits of nnnn is [ØØ] if the words are to be packed with no gaps, or [Ø1] if original words are always to start in the next unused 8-bit byte of the data stream. The second pair of digits contain a count of the number of bits in each word as transmitted by the sender.

Comments:

1. As an example, [ØØØC] would indicate that the bits of the original 12-bit words were packed thus:-

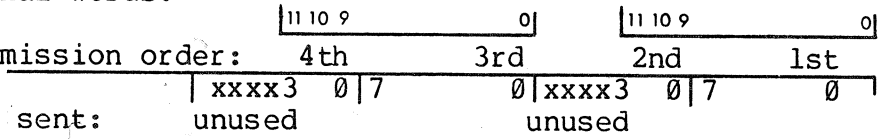
   Original Words:

   Bytes sent:

2. Whereas [Ø1ØC] would indicate that the bits of the original 12-bit words were packed with a residue, thus:-

   Original words:

3. The default setting for this parameter is [ØØØ8] or [Ø1Ø8], the effect of the two values being identical, i.e. the original binary data is assumed to be handled in 8-bit bytes.

4. Compression (if allowed by the option bit of the Facilities attribute (number [ØE], Section 5.12)) can be applied to the 8-bit bytes of the transmitted records after they have been packed with the binary data.

5. In either packing scheme, 16-bit words will be transmitted least significant byte first.

5.6 Maximum Record Size

Parameter: [Ø5] [2A] <N> if Take
or [Ø5] [AB] <N> if Give

Default value: [ØØFC] = 252 bytes

Function:

This parameter specifies the maximum size of record which may be transferred during the data transfer phase, before compression or division into sub-records.

Comments:

1. If the transfer is from Q to P, P uses this parameter to express the maximum size of records that it is prepared to receive. Q may reply with the known maximum record size in the file, if smaller.

2. If the transfer is from P to Q, this parameter expresses the maximum size of records that process P intends to transmit.

5.7 Maximum Transfer Size

Parameter: [Ø6] [2A] <N> if Take
or [Ø6] [AB] <N> if Give

Default value: [Ø4ØØ] = 1Ø24k = 1Ø48576 bytes

Function:

N specifies the maximum number of kilobytes (units of 1Ø24 bytes) of data that may be transferred during the data phase of the transfer. The size quoted is that before compression, and does not include any possible re-transmission of data.

Comments:

1. The recommended qualifier for use by P as sender sets the attribute value equal to the intended size of the transfer; if P is the receiver, the recommended qualifier yields a parameter indicating the maximum size it can accept.

5.8 Transfer Identifier

Parameter: [Ø8] [2A] <N>

Default value: [ØØØØ]

Function:

The parameter allows the process P to tag that particular file transfer.

Comments:

1. The Transfer Identifier can be used by each process for accounting or other purposes.

5.9 Acknowledgement Window

Parameter: [ØA] [2A] <N>

Default value: [ØØØ2]

Function:

N defines the maximum number of restart marks which, together with their associated data, the sender will transmit before receiving any acknowledgement from the receiver.

Specification:

N must lie in the range 1 to 255, the default value being 2. This parameter is ignored if mark acknowledgement is not being used. (See the Facilities attribute (number [ØE], Section 5.12).)

Comments:

1. The first restart mark is given with the Start of Data (SS) command. With the (default) setting of 2, and mark acknowledgement in use, the sender would cease transmitting immediately after two mark data (MS) commands have been sent, if no acknowledgement has been received.

2. During the data transfer phase the sender may be required to retransmit data, starting at the last mark acknowledged, or at any of the unacknowledged marks (if the Restart Request facility has been agreed - see the Facilities attribute (number [ØE], Section 5.12)).

3. To ensure a steady flow of data, the receiver should normally send an acknowledgement before the last permitted mark is received.

4. The parameter will normally be chosen by the sender, on the basis of knowledge of the algorithm that will be used for mark insertion (see Appendix V).

5.10 Initial Restart Mark

Parameter: [ØB] [2A] <N>

Default value: [ØØØØ]

Function:

N specifies the restart mark at which the transfer (re-transfer) is to start.

Comments:

1. The use of this parameter is described in detail in Section 2.5 and Appendix V.

2. The ability to continue a transfer by starting afresh at a non-zero mark can be indicated by the Facilities attribute (number [ØE], Section 5.12).

5.11 Minimum Time-out Interval

Parameter: [ØD] [AB] <N>

Default value: [Ø258] = 1Ø minutes

Function:

The parameter is used to specify the interval in seconds after which the file transfer will be unilaterally aborted and all knowledge of it subsequently denied.

Comments:

1. N should be chosen to exceed the longest acceptable delay taking into account network delays in both directions and the possibility of lower network protocol levels attempting retransmissions, encountering congested hosts, etc. This yields a value of some tens of minutes.

2. If Hold is to be used, the value of N must be longer than the longest anticipated Hold time.

3. For a more general description, see Sections 2.8 and 4.6.

4. The recommended qualifier for use by P allows Q to set a time-out value lower than the value quoted. In some cases, [AE] may be used to place a lower bound, for example, to allow for expected manual intervention.

5.12 Facilities

Parameter: [ØE] [AB] <nnnn>

Default value: [ØØØØ]

Function:

This parameter defines the set of facilities which may be used in the data transfer phase. These include data compression, restart mark acknowledgement, and various other facilities and options.

Specification:

Each binary digit of nnnn is used to indicate whether or not a defined option can be used. The following bits have been defined:-

Bit	Value	Option Enabled
Ø	[ØØØ1]	Compression of Level 2 data
1	[ØØØ2]	Later resumption of this transfer in new transfer possible
2	[ØØØ4]	Restart Requests permitted within this transfer
3	[ØØØ8]	Restart mark acknowledgement must occur
4	[ØØ1Ø]	Temporary Hold
5	[ØØ2Ø]	Parameters on the GO command
6-15	[FFCØ]	reserved

Comments:

1. The default value is [ØØØØ], that is operation without compression or restart mark acknowledgement. This allows primitive processes to avoid all 'advanced' facilities and options.

2. Compression, if selected, can be used with any of the codes selected in the Codes attribute (number [Ø2], Section 5.3), including binary. The use of compression is explained more fully in Section 3, and illustrated in Section 7.3.

3. The facility for inclusion of parameters on the GO command is provided to allow for experiment and further study.

5.13 State of Transfer

Parameter: [ØF] [2A] <nnnn>

Default value: [ØØØØ]

Function:

The parameter is used to convey information about the current state of the file transfer and, in particular, to provide additional explanation when the transfer has failed, or is being rejected.

Specification:

nnnn is regarded as two parts:

a) The most significant hexadecimal digit gives the general state of the transfer, as follows:

Number	State
Ø	Viable
1	Rejected
2	Terminated
3	Aborted

b) The least significant 3 hexadecimal digits give specific extra information about the state (e.g. 'fail number').

The coded values for nnnn are given in Appendix IV.

Comments:

1. If used with SFT, RPOS or GO commands, this parameter should have default value [ØØØØ] (Viable). Use with these commands is thus not normally useful.

5.14 Filename

Parameter: [40] [32] <string>

Default value: ""

Function:

The Filename is used to identify the file involved in the file transfer.

Comments:

1. The format of the Filename must be directly acceptable to the conceptual filestore at process Q. The question of standard filename formats is for further study.

5.15 Username

Parameter: [42] [32] <string>

Default value: ""

Function:

The Username is used to identify the username or owner of the file involved in the file transfer.

Comments:

1. The Username attribute may be known by different titles locally, for example Owner, Directory, Catalogue, or Clerk. It names the principal list in which the Filename is found as an entry.

5.16 Username Password

Parameter: [44] [32] <string>

Default value: ""

Function:

The parameter specifies the password necessary to permit access to the username or directory in which the file is located.

Comments:

1. The Username Password is the main password or key used to gain access to the filestore directory.

5.17 File Password

Parameter: [45] [32] <string>

Default value: ""

Function:

The parameter specifies the password or key necessary to permit access to the particular file named by the filename parameter.

5.18 Kinship

Parameter: [46] [32] <string>

Default value: ""

Function:

To indicate additional classifying information about the file involved in the transfer.

Comments:

1. This information has a wide variety of local titles, for example:- group, class, language code, type specifier, etc. It associates files of different names with the same purpose or function, or alternatively represents different sub-names for files which otherwise have the same name.

2. If P requests the current or preferred version of a file, Q should include the actual value of the attribute with the parameters of the reply command.

5.19 Account

Parameter: [4A] [32] <string>

Default value: ""

Function:

To indicate the name or number of the account to which the charges for the file transfer are to be made.

5.20 Account Password

Parameter: [4B] [32] <string>

Default value: ""

Function:

To specify the password necessary to permit access to the account for the purpose of charging for the file transfer.

5.21 Output Device Type

Parameter: [50] [32] <string>

Default value: "LP"

Function:

This parameter is used to specify the output device type for job output.

Specification:

This parameter is only meaningful with transfer modes [C001] or [4001] (Give or Take Job Output). Suggested device types are:

LP	Line Printer
CP	Card Punch

Comments:

1. The devices are assumed to use the standard default forms of the installation. There is no provision for special forms control in this protocol.

2. If a device (such as a plotter) expects data in a special format then this format must have been agreed between the two parties outside the protocol.

5.22 File Size

Parameter: [60] [2A] <N>

Default value: [0400] = 1024k = 1048576 bytes

Function:

Specifies the amount of filestore to be reserved for the file in kilobytes (units of 1024 bytes).

Comments:

1. The space reserved may be larger than the amount of data actually transferred (see Maximum Transfer Size attribute (number [06], Section 5.7)).

2. The detailed effect of the use of this parameter by the receiver is a subject for further study.

5.23 Operator Message

Parameter: [70] [3A] <string>

Default value: ""

Function:

To allow a text message to be conveyed with a command for output to an operator.

Comments:

1. The receiver should arrange logging in such a way that parameters [70] and [71] need not be sent with the same text string.

5.24 Monitor Message

Parameter: [71] [3A] <string>

Default value: ""

Function:

To allow a text message to be conveyed with a command for output to a logging or monitor file.

5.25 Special Options

Parameter: [8Ø] [3A] <string>

Default value: ""

Function:

This parameter allows the specification of special options known to the Q process.

Comments:

1. Implementations should avoid making extensive use of this parameter. It should only be used to access facilities not described by other attributes.

2. If it is believed that a requirement is of general interest, and that a standard attribute should be assigned for it, a member of the working group should be contacted (see Appendix VII).

3. The suggested qualifier for use with this parameter is [3A], which implies that the parameter sets the value of some option to be associated with an already selected item.

6. Data Transfer Control - Level 1 Commands

There are eight data transfer control commands. Four of these are used by the sender of the data, and four are used by the receiver.

Level 1 commands are always coded as two 8-bit bytes:-

<command number> <argument>

The start of the data transfer phase is signalled by a Level Ø GO command. An SS command must precede the transmission of the first data. The data stream consists of records encoded as described in Section 3.2. Level 1 commands are always transferred with a specially reserved zero header (which would otherwise precede a sub-record of zero length). During the data transfer phase the data passes from sender to receiver, until the final ES command. Only control commands can flow from the receiver to the sender. Section 2 contains further general description.

The commands are:

Command (Hex)	Name*	Use	Argument
4Ø	SS	Start of Data	mark number
41	MS	Mark Data	mark number
42	CS	Code Select	new code
43	ES	End of Data	status code
44	RR	Restart Request	mark number
45	MR	Mark Acknowledge	mark number
46	QR	Quit	status code
47	ER	End Acknowledge	status code

*The second letter of the short name indicates that the command is sent by Sender or Receiver.

Many of the commands occur as part of handshakes or short sequences, in which the receipt of one command prompts the transmission of another. In such circumstances, the earlier command is said to be outstanding until the consequent command is sent or received, or until the

command is rendered obsolete by a subsequent event. The commands have a precedence (often depending on their arguments), which determines the result when other commands are received while a command is outstanding. The newly received command may either render obsolete the outstanding one, or the new command may be discarded as being less important. In particular, amongst the group of commands ES, QR and ER, the sub-group OK is superseded by Hold, Hold by Error, and Error by Abort (consistent with the argument numbering).

The Abort form terminates the data transfer phase and so can never be outstanding.

A command is said to match the one it follows if they are both from the same sub-group. Often the arguments will be identical,but this may not be so if, for example, multiple errors are detected.

The commands and their normal causes and consequences are shown below.

Name	Follows	Followed by	Unsolicited Use
SS	RR	--	At Start of Transfer
MS	--	--	To Mark Point in Data
CS	--	--	To Select a Code
ES	QR	ER	To Terminate Transfer
RR	--	SS	To Request Restart
MR	--	--	To Acknowledge Data
QR	--	ES	To Terminate or Suspend Transfer
ER	ES	--	--

6.1 SS - Start of Data

Command: [40] <mark number>

Function:

The Start of Data (SS) command is used by the sender to herald the start of the transfer of data. The argument for this initial use is the least significant 8 bits of the value of the Initial Restart Mark attribute (number [ØB], Section 5.1Ø), and defines the first restart point. This argument will be [ØØ] unless the file transfer is a retry of a former partly successful attempt.

The SS command is also sent in response to a Restart Request (RR) command from the receiver. In this second case it will quote the mark number used in the RR command.

A Start of Data command forces the selected code to be IA5 with any parity, as if CS[ØØ] had been sent.

Comments:

1. The SS command is only legal in response to an RR command, or as the first command in Level 1.

2. If the code in use is other than IA5 with any parity, a Code Select (CS) command must follow the SS command.

6.2 MS - Mark Data

Command: [41] <mark number>

Function:

The sender inserts a Mark Data command into the data stream, to identify the point reached. The positioning of such marks is at the sender's discrection. These marks define the points from which re-transmission may be requested, if restarts are allowed by the Facilities attribute (number [ØE], Section 5.12). Such a restart request may be made at any time until a subsequent mark has been acknowledged. This topic is discussed in Section 2.4.

Restart marks are consecutively numbered; the argument of an MS command is the least significant 8 bits of the mark number. Thus the argument is obtained from that of the preceding MS or SS command by incrementing modulo 256.

Comments:

1. It is recommended that restart marks should be put at every page of data in the case of line printer files.

2. The sender should not transmit data after sending the last mark allowed by the Acknowledgement Window attribute (number [ØA], Section 5.9), until some furtner acknowledgement has been received. The receiver must send such an acknowledgement within the time-out period.

3. Marks should not be placed between the sub-records of a multi-part record.

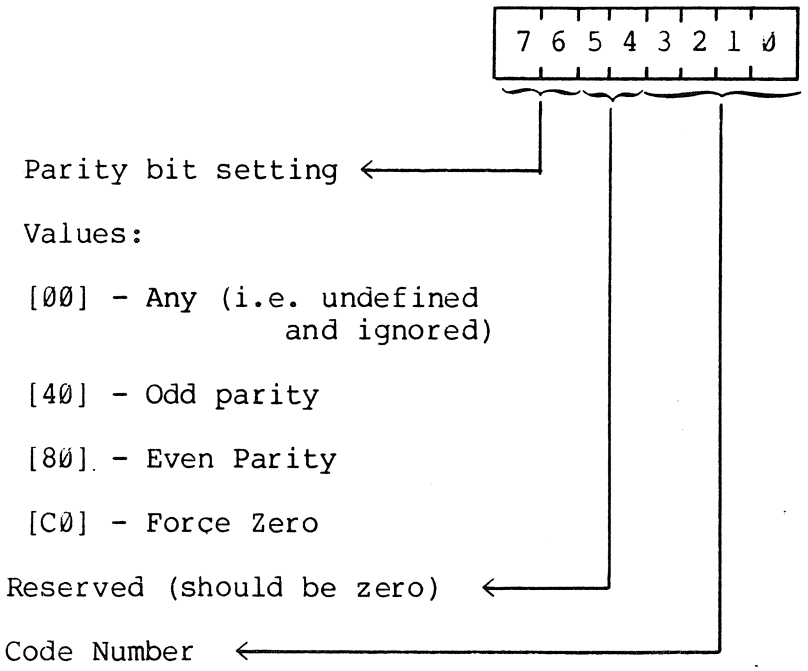
4. The positioning of restart marks is discussed in Appendix V.

6.3 CS - Code Select

Command: [42] <new code>

Function:

The sender uses this command to select a previously agreed code to be used in the subsequent data transmission. The new code is derived from the value of the Codes attribute (number [Ø2], Section 5.3), the choice being indicated by a number corresponding to the code's bit number in the attribute.



Comments:

1. After any SS command the code is IA5 with any parity, as if CS[ØØ] had been sent.

2. The parity bit setting is unimportant when EBCDIC or binary is selected.

3. Code changes should not be made between the sub-records of a multi-part record.

6.4 ES - End of Data

Command: [43] <status code>

Function:

The ES command is transmitted by the sender, either to signify that the previous record was the last (normally the end of file), or in response to a QR command. The status code gives the sender's view of the state of the transfer.

There are four groups of ES commands. These are, in order of increasing precedence:

a) OK - ES(OK), which is used by the sender to indicate that end of data has been reached. This is the beginning of the final end of file handshake for a sucessful transfer. An ES(OK) can also be sent in response to a QR(OK). It requests an ER(OK) to complete the transfer.

   Arguments: [ØØ]

b) Hold - ES(H), which is used in response to a QR(H) to signal that the sender is entering the Hold state (see Section 2.6), and will refrain from data transmission until the Hold state is ended by an ER(H), a QR of higher precedence, or a transport service reset. An ES(H) can only be used in response to an outstanding QR(H); an ES(H) should be ignored if no QR(H) is outstanding.

   Arguments: [1Ø]

c) Error - ES(E), which is used in response to a QR(E), or to signal that a premature termination is necessary because of some error detected by the sender. It requests a matching ER(E) to complete the data transfer phase.

   Arguments:
   [2Ø] Receiver error, retry possible.
   [21] Receiver error, no retry possible.
   [22] Protocol error detected by receiver.
   [23] GO not acceptable to receiver.

   [28] Sender error, retry possible.
   [29] Sender error, no retry possible.
   [2A] Protocol error detected by sender.
   [2B] GO not acceptable to sender.

d) Abort - ES(A), which is used to cause immediate termination of the data transfer phase, after a time out. No response is required.

Arguments: [30] Timed out awaiting an MR.
[31] Timed out awaiting an RR after a Reset.
[32] Timed out awaiting an ER(OK), after sending an ES(OK).
[33] Timed out awaiting an ER(E), after sending an ES(E).

In all cases, an ES command supersedes any outstanding lower precedence (weaker) one or any weaker QR. The receiver should ignore an ES if there is a higher precedence (stronger) QR outstanding. An ES command is outstanding until acknowledged by an ER command with identical argument, or until superseded by a stronger QR or until cancelled by a Reset. An outstanding ES(OK) is cancelled by an RR command.

Comments:

1. The sender may still receive an MR or QR command after sending an ES, until an ER is received. These must be acted upon in the normal manner.

2. If the transfer is to terminate successfully, an ES(OK) which has been cancelled by an RR or a QR(H) must be re-transmitted.

3. An ES(A) will terminate the data transfer phase. If the sender is the initiator of the transfer, it should also send a Level 0 STOP command.

6.5 RR - Restart Request

Command: [44] <mark number>

Function:

The receiver sends a Restart Request to cause re-transmission from the quoted restart mark. This mark must be between the restart mark last acknowledged by the receiver and the mark last received from the sender, inclusive.

A Restart Request is outstanding until acknowledged by a matching Start of Data command (or cancelled by errors or transport service reset). An RR command can be sent whenever one is not outstanding, but the command can only be used if the appropriate option has been enabled in the Facilities attribute (number [ØE], Section 5.12).

An RR command cancels any outstanding ES(OK).

Comments:

1. An RR command can be cancelled by a transport service reset, any QR (except QR(H)), an ES(E) or an ES(A).

2. A Restart Request does not have any acknowledgement function.

3. The period during which a Restart Request is outstanding should be monitored, and suitable time-out action taken.

4. It is not possible to specify a restart point which is ahead of the sender's last mark.

5. The Acknowledgement Window ensures that there are at most 255 unacknowledged marks at any time; thus there is no ambiguity in using only the least significant 8 bits of the mark number in the MR and RR commands.

6. An RR command should be sent if the transfer is to continue after a transport service reset, unless data flow has been suspended by a temporary Hold.

6.6 MR - Mark Acknowledge

Command: [45] <mark number>

Function:

The receiver uses the Mark Acknowledge command to acknowledge and confirm the satisfactory receipt of all data records up to the quoted restart point. The sender will not be requested to retransmit from marks earlier than the last mark acknowledged.

The argument gives the least significant 8 bits of the mark to be acknowledged. This mark must lie between the last mark acknowledged and the last mark received, inclusive.

Mark Acknowledge commands must be used if required by the value of the Facilities attribute (number [ØE], Section 5.12). These commands can be used at any time; in particular one should be sent following a transport service reset if the transfer is to continue.

Comments:

1. The last mark acknowledged is initially set to the value of the Initial Restart Mark attribute (number [ØB], Section 5.1Ø). Thus the mark number in the initial SS command at the start of the transfer needs no acknowledgement. (There is no preceding data to acknowledge.)

2. It is permissible to acknowledge the same mark repeatedly, but it is not subsequently permissible to acknowledge an earlier mark.

3. The last mark acknowledged should be repeated after a transport service reset, in order to prevent possible deadlocks that might occur if an acknowledgement had been lost.

6.7 QR - Quit

Command: [46] <status code>

Function:

The Quit command is used by the receiver, either to indicate that it is unable to continue the transfer, or to request a temporary Hold. There are four groups of QR commands which are, in order of increasing precedence:

a) OK - QR(OK), which is used when sufficient data has been received, or when an ES(OK) has been received but cancelled by a transport service reset. It requests the transmission of an ES(OK). QR(OK) may not be used while an RR command is outstanding.

   Arguments: [ØØ]

b) Hold - QR(H), which is a request for a temporary Hold. It solicits an ES(H), to mark the start of the Hold state. The Hold can be ended by the receiver sending an ER(H) to request the sender to resume transmission. An ER(H) can also be sent while the QR(H) is outstanding, and will supersede it. This command can only be used if the appropriate option has been enabled in the Facilities attribute (number [ØE], Section 5.12).

   Arguments: [1Ø]

c) Error - QR(E), which is used by the receiver to cause premature termination after an error has occurred. The argument indicates the class of error. The QR(E) solicits a matching ES(E).

   Arguments:
   [2Ø] Receiver error, retry possible.
   [21] Receiver error, no retry possible.
   [22] Protocol error detected by receiver.
   [23] GO not acceptable to receiver.

d) Abort - QR(A), which requests an immediate termination, without a response from the sender. It can be used after a time-out has occurred.

   Arguments:
   [3Ø] Timed out, awaiting data.
   [31] Timed out, after sending RR, awaiting SS.
   [32] Timed out, after sending QR(OK), awaiting ES(OK).
   [33] Timed out, after sending QR(H), awaiting ES(H).
   [34] Timed out, after sending QR(E), awaiting ES(E).
   [35] Timed out, after GO, awaiting first SS.

A QR supersedes any weaker QR, and cancels any outstanding weaker ES. It remains outstanding until replied to by a matching ES, or until cancelled by a stronger ES or a Reset. A QR received while a matching or stronger ES is outstanding should be ignored. An outstanding Hold request QR(H) is superseded by an ER(H).

Comments:

1. The Hold state introduced by QR(H) should not last for longer than the limit given by the Minimum Time-out Interval attribute (number [ØD], Section 5.11).

2. A QR(A) command will terminate the data transfer phase. If the receiver is the initiator of the transfer (P), it should also send a Level Ø STOP command.

3. QR(H) can be used to re-establish a Hold state after a transport service reset. An ES(H) reply is still required in this case.

6.8 ER - End Acknowledge

Command: [47] <status code>

Function:

The End Acknowledge command is used by the receiver to confirm the receipt of an ES command, or to release a temporary Hold. There are three groups of ER commands which are, in order of precedence:

a) OK - ER(OK), which indicates that the data transfer phase has ended without error. This is the normal end of the data transfer phase.

   Arguments: [00]

b) Hold - ER(H), which is used to cancel an established Hold, or a request for a Hold. This command should only be used while an ES(H) or QR(H) command is outstanding.

   Arguments: [10]

c) Error - ER(E), which indicates that the data transfer phase has terminated in error.

   Arguments:
   [20] Receiver error, retry possible.
   [21] Receiver error, no retry possible.
   [22] Protocol error detected by receiver.
   [23] GO not acceptable to receiver.

   [28] Sender error, retry possible.
   [29] Sender error, no retry possible.
   [2A] Protocol error detected by sender.
   [2B] GO not acceptable to sender.

An ER command satisfies an outstanding ES command with identical argument. An ER(H) can supersede an outstanding QR(H) command. The ER command ends a sequence, and so cannot be outstanding.

Comments:

1. The receiver must send back an ER for each and every ES received (except ES(A), when it is too late), unless the ES is superseded by some other command.

2. The receiver may send an RR command before an ER(H) if it wishes to resume from an available mark after Hold.

3. A valid ER(OK) or ER(E) terminates the data transfer phase and therefore supersedes any outstanding Level 1 command.

7. Examples and Illustrations

The following examples of the File Transfer Protocol seek to illustrate some of the main sequences that could occur in normal use. It is by no means an exhaustive set.

The terminology set out in Section 1 has been used throughout. In particular:

a) where the 'Mode of Access' implies that the data forming the body of the file will move from the process P into the filestore mapped by process Q, then this is termed a 'take', and

b) where the file data will move out of the filestore at Q this is termed a 'give'.

7.1 Data Transfer from P to Q (Take)

7.1.1 Process P sends a command to start the file transfer (SFT) with associated parameters. Process Q may choose to reject the proposed file transfer, in which case a negative reply (RNEG) will be returned, quoting a reason for the failure. Process P sends a STOP finally terminating the transfer. Figure 8 illustrates this situation.

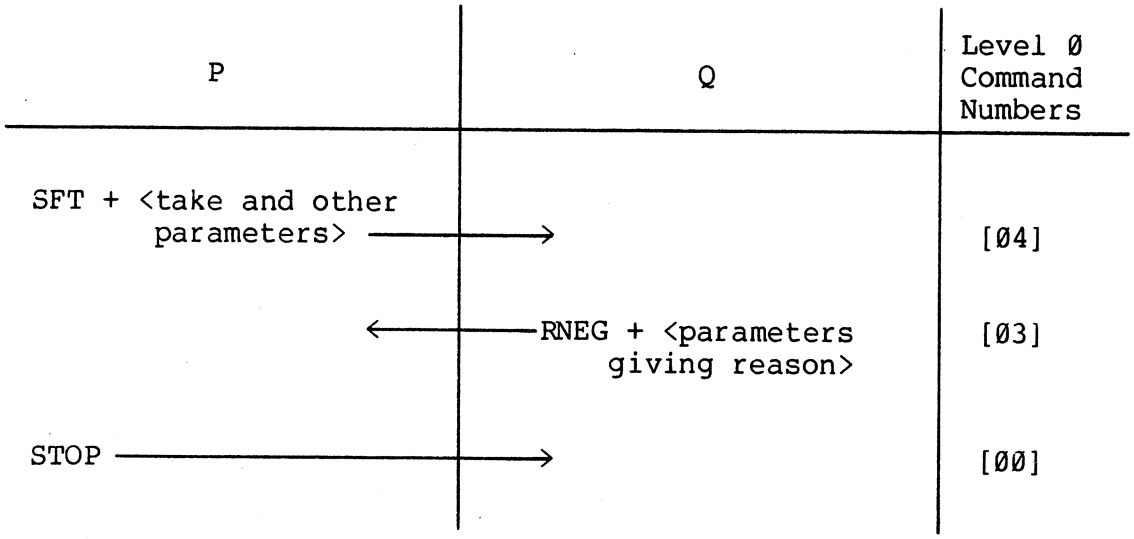


Fig. 8 File transfer rejected by process Q

7.1.2 Alternatively, process Q may be willing to accept the proposed transfer, possibly with adjusted values for some or all of the attributes. In this case, a positive reply (RPOS) is returned, together with any parameters, giving the values indicated as needed by the SFT command (e.g. 'Facilities').

Process P may be content to accept the restricted values (see Section 7.1.3 below) or, alternatively, may find them inadequate. In the latter case, process P sends a STOP to process Q, giving a failure reason. The parameters sent should include at least the State of Transfer attribute (number [ØF], Section 5.13) and an Operator or Monitor Message (numbers [7Ø] or [71], Sections 5.23 or 5.24). Figure 9 summarises this interchange.

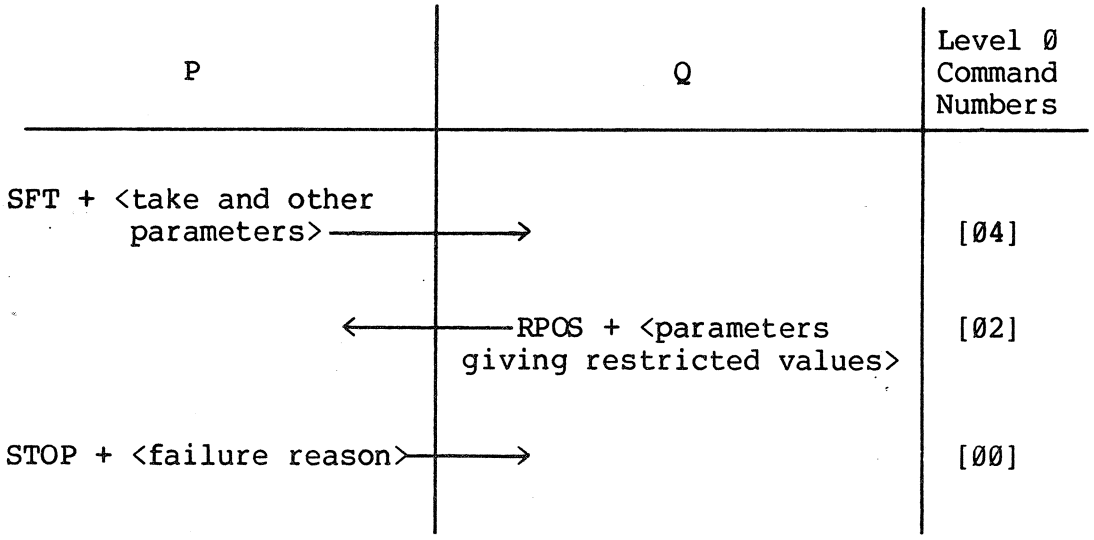


Fig. 9 Adjusted values unacceptable to process P

7.1.3 If the originally quoted set, or reduced set, of attributes values is acceptable, process P is now able to start transmitting data, using Level 1 data transfer control commands. Process P becomes the 'sending' end, Q the 'receiving' end. The receiving end can exercise control by transmitting commands as follows:-

a) A command (MR) to acknowledge the acceptance of data, up to the restart point whose number is quoted.

b) A request (QR) to process P to abort the file transfer, plus information on whether or not to retry. Process Q can then ignore all incoming data until End of Data (ES) command (with the appropriate status) is received from process P. Process Q then confirms with an End of Data (ER) command.

Figure 1Ø illustrates this aborted transfer.

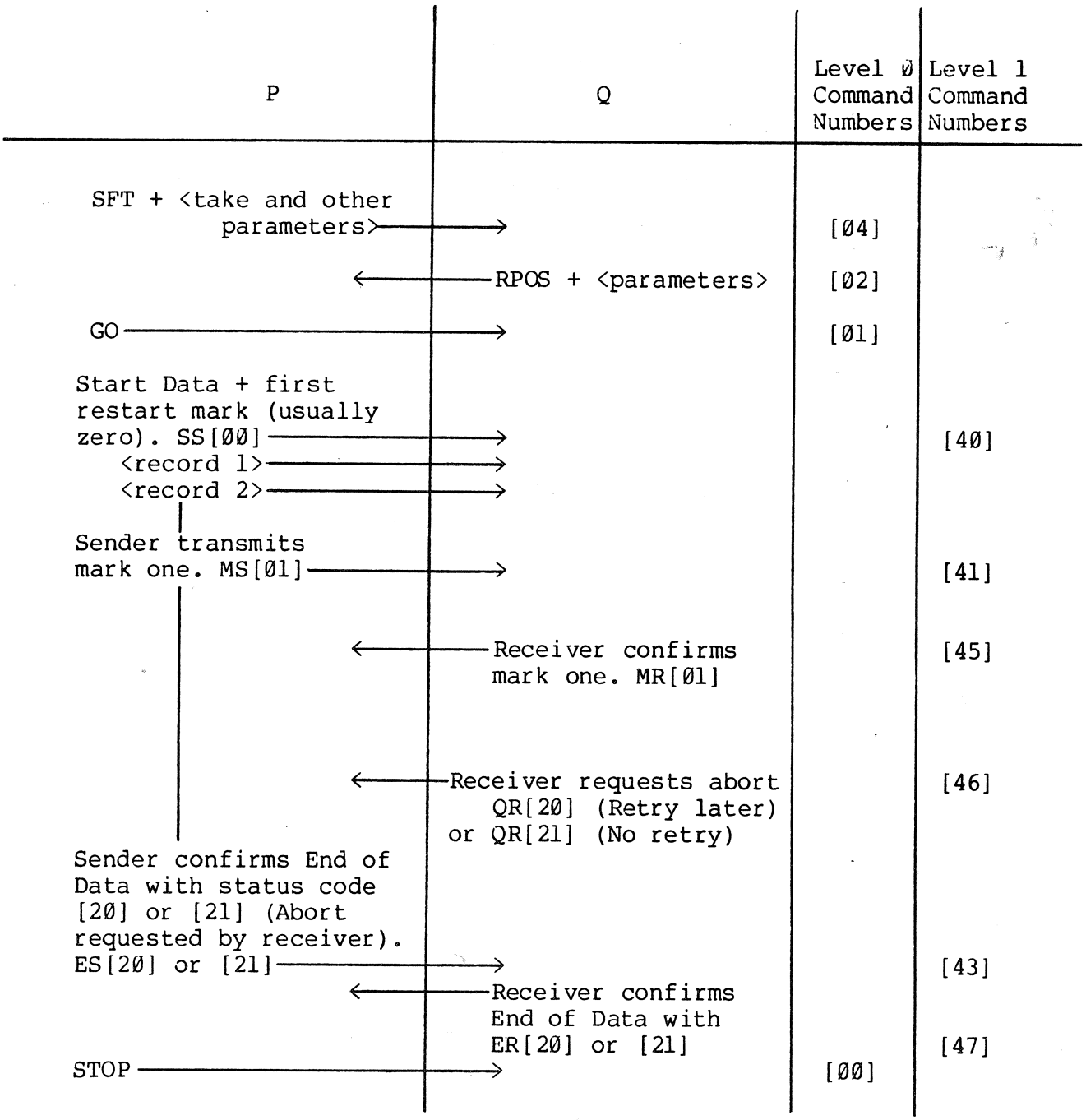


Fig. 1Ø File transfer aborted by process Q

7.1.4 At the end of the data transfer an 'End of Data' command is transmitted to process Q. Process Q confirms 'End of Data' (ER) to process P. This last interaction is necessary to ensure that data has been received satisfactorily and to check that no abort or restart requests are in transmission from process Q. Process P finally terminates the interaction with a STOP, as before. Figure 11 illustrates this interchange.

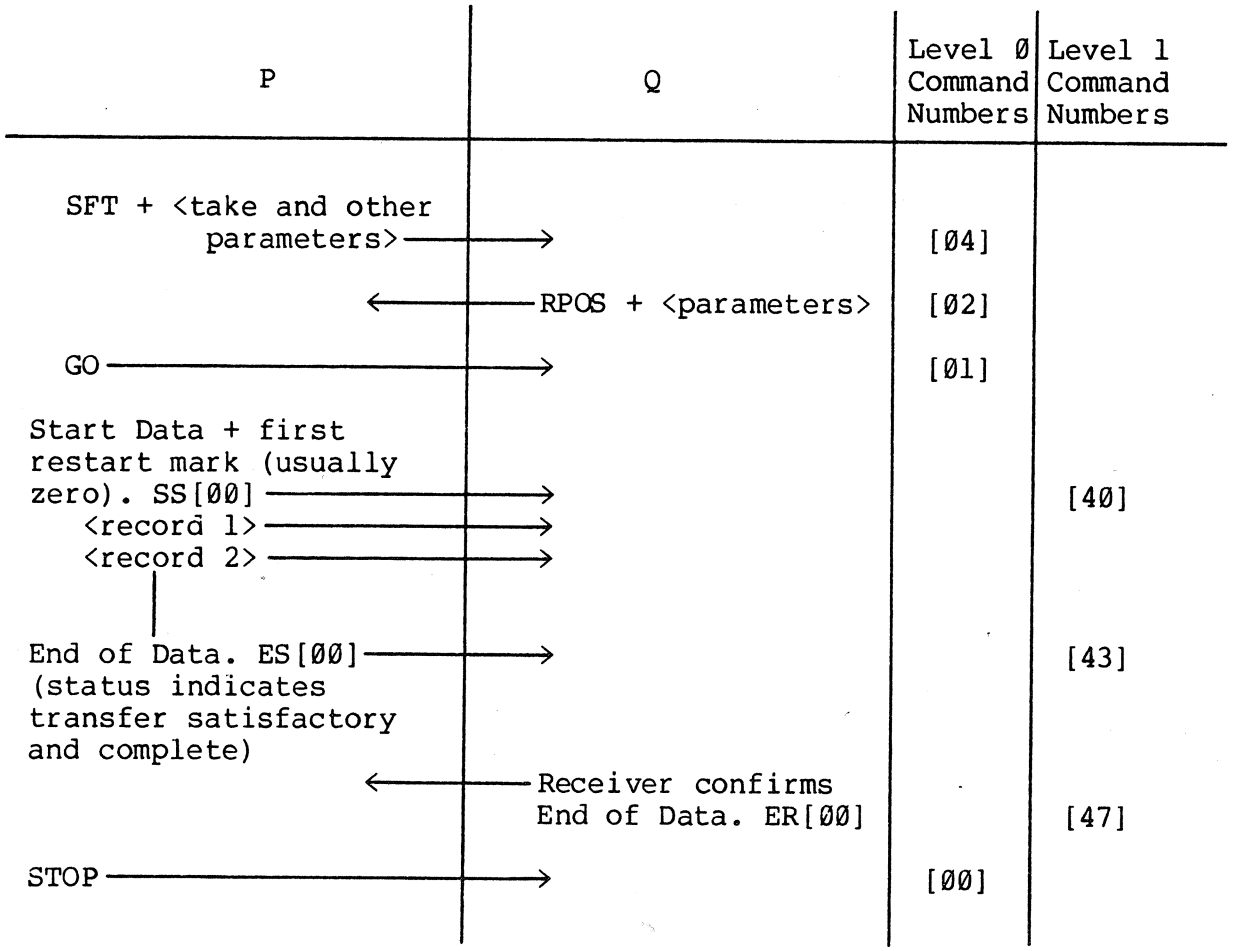


Fig. 11 Complete data transfer from P to Q

7.2 Data Transfer from Q to P (Give)

7.2.1 Process P sends the command to start the file transfer (SFT) with its associated parameters. If at this point, process Q chooses to reject the proposed file transfer, a negative reply (RNEG) is returned, quoting a reason for the failure, and thereby terminating the interaction. Figure 12 illustrates this interchange. One reason for a failure at this point might be that the file quoted could not be found by process Q. If process Q can only use a sub-set of the quoted values then the transfer can proceed as detailed in Section 7.2.2 below. If, on the other hand, the values quoted specifically exclude a value required by process Q, then the transfer cannot proceed and must be terminated.

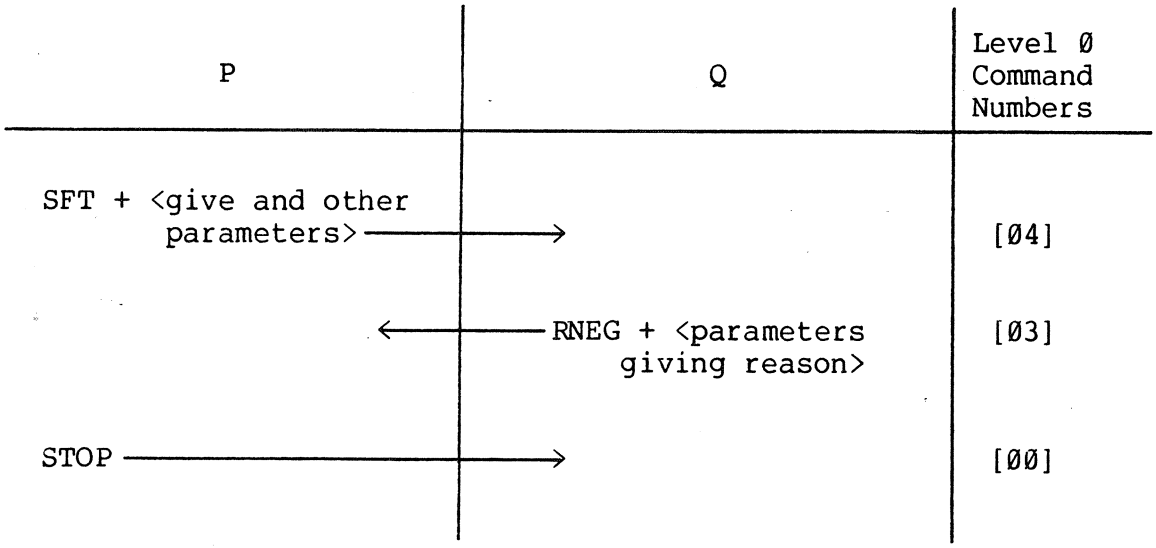


Fig. 12 File Transfer rejected by process Q

7.2.2 If the transfer control attribute values (or a sub-set of them) are acceptable to process Q and the file referred to has been found, then it sends a positive reply (RPOS) to process P. If process P agrees that the intended file is being accessed then it sends GO to begin the data transfer phase. Process Q becomes the 'sending' end and process P the 'receiving' end. Process P can exercise control by transmitting commands as follows:

a) A command (MR) to acknowledge the acceptance of data up to the restart point whose number is quoted.

b) A request (QR) to process Q to abort the file transfer plus information on whether or not to retry. Process P (the receiving end) can ignore all data until 'End of Data' (ES) is received from process Q. Process P then confirms 'End of Data' (ER).

Figure 13 illustrates this aborted transfer.

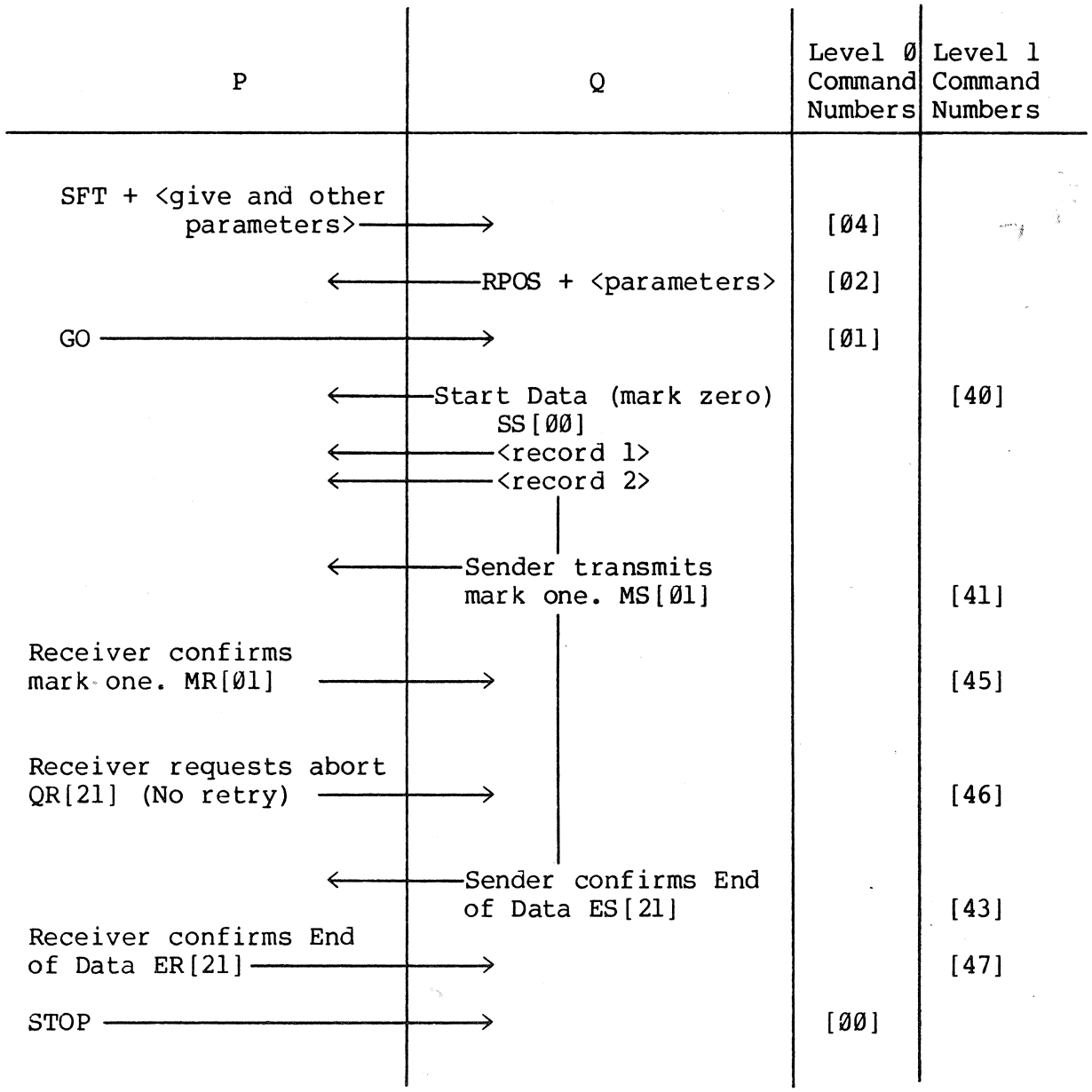


Fig. 13 File transfer aborted by process P

7.2.3 At the end of the file an 'End of Data' command (ES) is transmitted to process P. Process P confirms 'End of Data' (ER) to process Q. Again, this last interaction is necessary to ensure that the data has been received satisfactorily, and to check that there are no abort or restart requests in transmission. Process P finally terminates the interaction with a STOP, as before. Figure 14 illustrates this interchange.

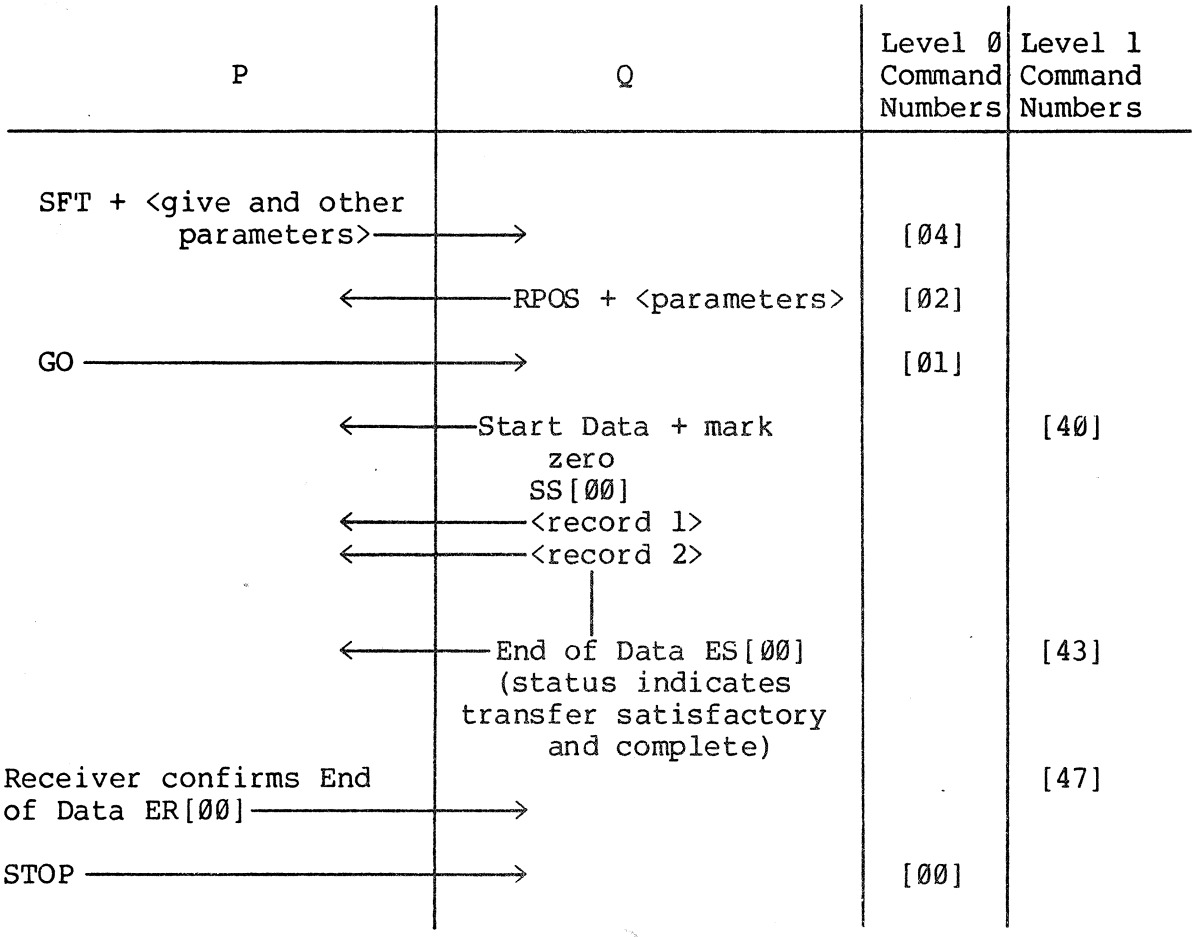


Fig. 14 Complete data transfer from Q to P

7.2.4 The final example demonstrates the use of restart facilities. Here the receiver gets into difficulties (e.g. the paper runs out on a line-printer) and loses some data. The receiver then requests a restart from a previously unacknowledged restart mark. Figure 15 illustrates the exchanges.

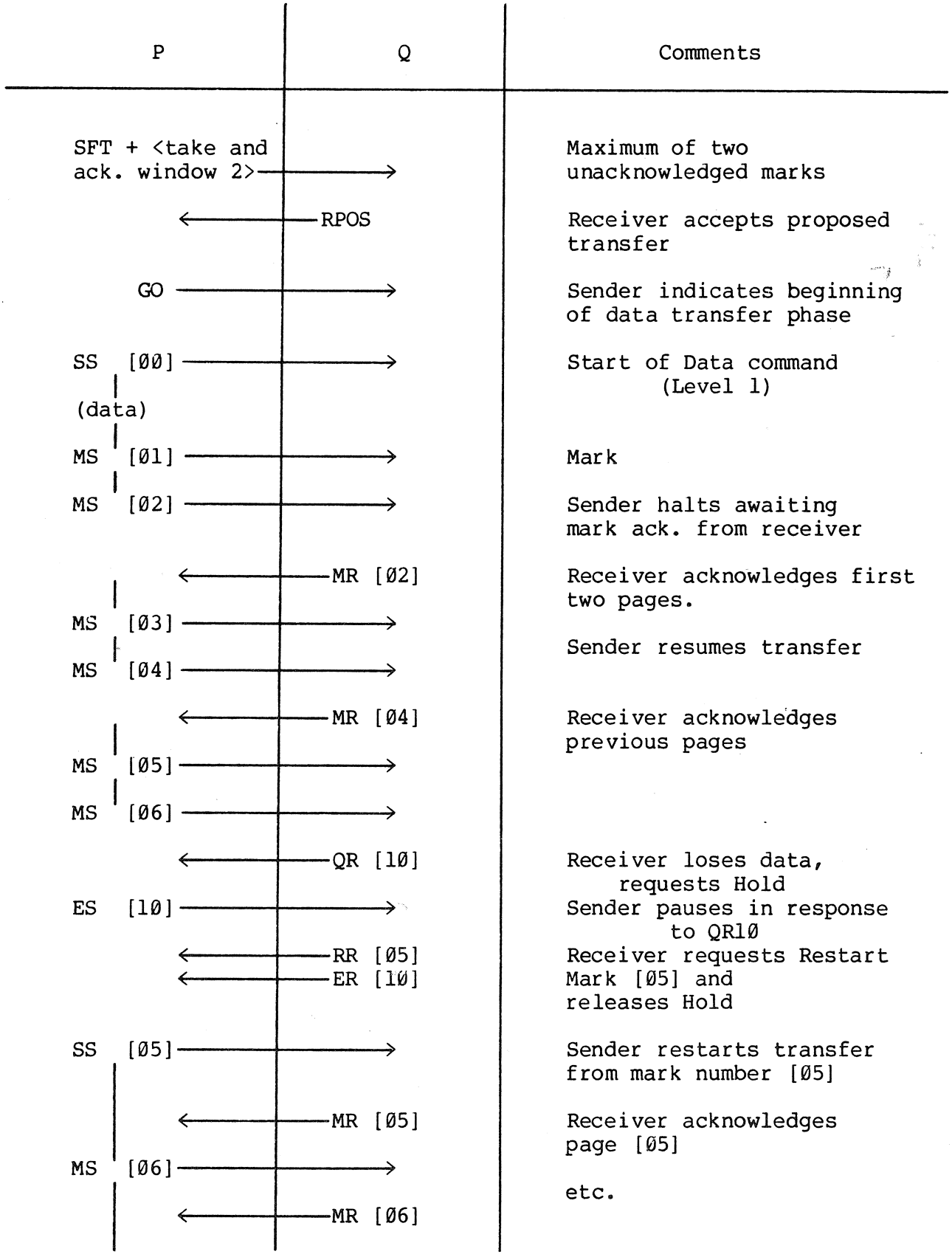


Fig. 15 File transfer showing the use of restart facilities.

7.3 Examples of Record Formats

a) To transmit two records:-

ABCDE followed by HIJKLMNOP

the bytes in the data stream would be:-

```
[85]ABCDE      total of 6 bytes
[89]HIJKLMNOP total of 1Ø bytes
```

b) To transmit the record:-

ABBBBCCCCCC

the bytes in the data stream could be:-

```
   [8B]ABBBBCCCCCC   total of 12 bytes
or [Ø1]A[44]B[C6]C   total of 6 bytes - compressed
```

c) To transmit the record:-

AA

the bytes of the data stream could be:-

```
   [82]AA   total of 3 bytes
or [C2]A    total of 2 bytes - compressed
```

d) The following technique is used to transmit a record whose length is greater than 63 bytes. Assuming the record is 128 bytes long and cannot be compressed:-

[3F]..1st 63 bytes..[3F]..2nd 63 bytes..[82]..last 2 bytes

Appendices

Appendix I

Summary of Level Ø commands

Command (Hex)	Purpose	Name	Used by	Section
ØØ	End of File Transfer	STOP	P	4.5
Ø1	Start Data Transfer	GO	P	4.4
Ø2	Reply-positive	RPOS	Q	4.2
Ø3	Reply-negative	RNEG	Q	4.3
Ø4	Start File Transfer	SFT	P	4.1

Appendix II

Summary of Level 1 Commands

Command (Hex)	Description	Argument	Section
40	SS - Start of Data	mark number	6.1
41	MS - Mark Data	mark number	6.2
42	CS - Code Select	new code	6.3
43	ES - End of Data	status code	6.4
44	RR - Restart Request	mark number	6.5
45	MR - Mark Acknowledge	mark number	6.6
46	QR - Quit	status code	6.7
47	ER - End Acknowledge	status code	6.8

Appendix III

Summary of Attributes and Default Values

Code No (Hex)	Attribute Name	Default value	Format	Section
	[Transfer Control Group]			
ØØ	Protocol Identification	[ØØØØ]	Binary	5.1
Ø1	Mode of Access	[ØØØØ]	Binary	5.2
Ø2	Codes	[1ØØ1]	Binary	5.3
Ø3	Format Effectors	[ØØØ1]	Binary	5.4
Ø4	Binary Mapping	[ØØØ8]	Binary	5.5
Ø5	Maximum Record Size	[ØØFC]	Binary	5.6
Ø6	Maximum Transfer Size	[Ø4ØØ]	Binary	5.7
Ø8	Transfer Identifier	[ØØØØ]	Binary	5.8
ØA	Acknowledgement Window	[ØØØ2]	Binary	5.9
ØB	Initial Restart Mark	[ØØØØ]	Binary	5.1Ø
ØD	Min. Time-out Interval	[Ø258]	Binary	5.11
ØE	Facilities	[ØØØØ]	Binary	5.12
ØF	State of Transfer	[ØØØØ]	Binary	5.13
	[Identification Group]			
4Ø	Filename	""	Alpha	5.14
42	Username	""	Alpha	5.15
44	Username Password	""	Alpha	5.16
45	Filename Password	""	Alpha	5.17
46	Kinship	""	Alpha	5.18
4A	Account	""	Alpha	5.19
4B	Account Password	""	Alpha	5.2Ø
	[Secondary Info Group]			
5Ø	Output Device Type	"LP"	Alpha	5.21
	[Scheduling Group]			
6Ø	File Size	[Ø4ØØ]	Binary	5.22
	[Information Group]			
7Ø	Operator Message	""	Alpha	5.23
71	Monitor Message	""	Alpha	5.24
	[Specials Group]			
8Ø	Special Options	""	Alpha	5.25

Appendix IV

Code Values for State of Transfer Attribute

Code No. (Hex)	Meaning
	VIABLE
0000	Acceptable and Satisfactory
	REJECTED
1000	Unspecific Transfer
1001	See text message supplied
1002	Unacceptable transfer control attribute settings
1010	Unspecific Filestore
1011	File not found or does not exist
1012	No access to file quoted
1013	File wrong type
1014	File not available or offline
1015	Username unknown
1016	Username Password not quoted or quoted incorrectly
1017	Account unknown
1018	Account Password not quoted or quoted incorrectly
1019	No money left
101A	File size quoted too big
101B	Output device unknown or unavailable
	TERMINATED
2000	Satisfactory and Complete
2001	Problem - see text message
	ABORTED
3000	Satisfactory and Incomplete - no retry required
3001	Receiver problems - retry possible
3002	Receiver problems - no retry possible
3003	Sender problems - retry possible
3004	Sender problems - no retry possible
3005	Time-out matured

Appendix V

Implementation Notes

The notes collected here seek to clarify some aspects of the protocol by reference to possible implementations. They should not, however, be considered as constraints on the way in which the protocol is to be used, or as forming any part of the protocol definition.

1. Transport Service

The protocol assumes that there is available a pair of synchronised byte streams (see Section 1.4). This will have been established before the FTP dialogue begins, and may be provided by a newly created virtual call, or may be a permanent circuit. Any addressing or routing required to identify or create the path will have been handled outside this protocol.

The synchronisation of the byte streams at a known initial point is required so that the record structure used can be interpreted and commands identified. For a virtual call, the initial condition would be a consequence of setting up the call, by a call establishment message and its reply. For a permanently established link, a specific synchronise command (reset) and its response could be used, or it might be known that the previous transmission had left the link in a tidy state.

Similarly, after a transport service detected error, or a deliberately generated reset, the synchronisation would be re-established by sending a suitable chain of messages around the loop formed by the two communication paths, as when establishing the initial state.

After the end of the transfer, the link may be closed, or may remain for future use. If the close operation is destructive (as in EPSS), then the close must normally be issued by Q, after the STOP has been received.

2. The Restart Mechanism

a) Styles of Use of the Restart Mechanism

The restart mechanism may be used when one or other terminal, or the communication link between them, is not sufficiently reliable for the intended application. It provides an end-to-end mechanism for guaranteeing the safe delivery and storage of the data. For the mechanism to be effective, data must not be acknowledged until it has been safely stored. The particular application, and the way in which the data

is disposed of by the receiver, determine what precautions are needed before data may be regarded as safely accepted and stored. If the data is being presented to a human operator (perhaps via a visual display), the data may be considered to be accepted when the operator acknowledges the data (perhaps by clearing the screen of the display). If the data is being written to a disc or other permanent storage, the data may be acknowledged as soon as it has been written and any associated directory updates have been made. A third example is provided by data sent to a line printer. In this case, there is no clear indication of when the data is safe; illegible print, or a paper wreck, may only be noticed by the operator after a considerable number of further pages have been printed. Thus it will be desirable to delay acknowledgements for as long as the Acknowledgement Window value will permit, without jeopardising the smooth flow of data.

Conversely, the requirements of the sender can be considered. If the data source is a disc, it will normally be easy to retransmit arbitrarily much of the file, so that a large Acknowledgement Window, or a large distance between marks, will present little problem to the sender. If, however, the data source is volatile (such a card or paper-tape reader, or a gateway to another network), the sender will have to keep copies of all unacknowledged data, and will therefore wish both the Acknowledgement Window and the spacing of marks to be small.

When the various combinations of sender and receiver are considered, it can be seen that the style of use of the restart mechanism is determined primarily by the weaker (less intelligent or less reliable) end of the link. If a transfer is from disc to disc, each end has reasonably high reliability; the restart mechanism is relatively unimportant, and the associated attribute values are not critical. For a disc to printer transfer, the attribute values will be determined largely by the requirements at the printer; for a card reader to disc transfer, the values will be determined largely by the source constraints. While the protocol would allow a card reader to printer transfer, this style of use is not usually satisfactory, since the constraints from the two ends are not compatible.

b) Positioning of Restart Marks

There are several considerations in the positioning of restart marks, according to the required style of use, and according to the way in which Restart Request commands are generated.

i) If Restart Requests are generated in response to operator actions (as in the printer example), it will be necessary to place marks at known points related to the structure of the file (such as at page boundaries). The file structure must be understood by both sender and receiver.

ii) If restart marks correspond to divisions in the structure of the input file, such as at disc block or track boundaries, the sender's job may be simplified considerably when repositioning for a restart.

iii) Where the sender has limited storage (as in the card reader example), it will wish to use the acknowledgement mechanism to limit the amount of data which has to be saved, and will place marks as closely as possible.

iv) The 'continue new transfer' facility can only refer to the first 65535 restart marks. This is not normally a significant restriction, if the marks are well spaced, but might become restrictive if the marks were placed after every record.

c) Acknowledgement Strategy

In order to implement possible Restart Requests, the sender must maintain an index giving, for each mark from the last acknowledged up to the last sent, details of where the corresponding data is stored. The Acknowledgement Window determines the maximum size that this index can have; thus, in general, the sender has an interest in keeping the Acknowledgement Window as small as possible, consistent with the smooth flow of data, and with the receiver's requirements. When the sender is storing data as well as an index, as in the card reader example, there will be a correspondingly greater incentive to keep the Acknowledgement Window small.

In some cases, the sender will not need to store the index explicitly, but will be able to compute entries as required. This will apply to technique (ii) in (b) above, and may also apply to technique (i).

These considerations, and the remarks in section (a), can be applied to the choice of Acknowledgement Window in particular examples:

i) for a disc to disc transfer, the default value of 2 will ensure smooth flow of data, and there are no other constraints to satisfy. Acknowledgements should be sent at the earliest safe opportunity.

ii) for a disc to printer transfer, where an explicit index is not needed, the maximum value of 255 can be used, and acknowledgements should be delayed until nearly the full number are outstanding.

iii) for a disc to printer transfer, where an explicit index is needed, the same strategy should be used, but the Acknowledgement Window should be limited to 10-20.

iv) for a card reader to disc transfer, where the mark spacing is very small, the value will usually need to be increased above its default to ensure smooth flow. The sender's requirements indicate that a value of about 5 would be suitable, with acknowledgements sent at the earliest opportunity.

d) Algorithms for Computing Mark Numbers

If the 'continue in new transfer' facility is to be used, it is necessary to record and manipulate full mark numbers, rather than merely the least significant 8 bits as provided by the MR and RR commands. The following algorithms can be used to compute the full mark numbers.

The sender maintains two counts:

LMS = last mark sent,
LMA = last mark acknowledged.

Both these counts are initialised to the Initial Restart Mark value. LMS is incremented whenever an MS command is to be sent; its least significant 8 bits are used as the argument of the MS command.
If LMS $\geq$ LMA + Acknowledgement Window, data transmission then ceases until a suitable acknowledgement is received.

When an MR command is received, LMA is updated by the formula:

LMA := LMA + ((argument-LMA) mod 256) .

When an RR command is received, LMS is updated similarly:

LMS := LMA + ((argument-LMA) mod 256) ,

and the new value of LMS determines the data to be retransmitted. In both cases, there is a protocol violation if the bracketed expression exceeds the Acknowledgement Window.

Appendix VI

State Transition Tables

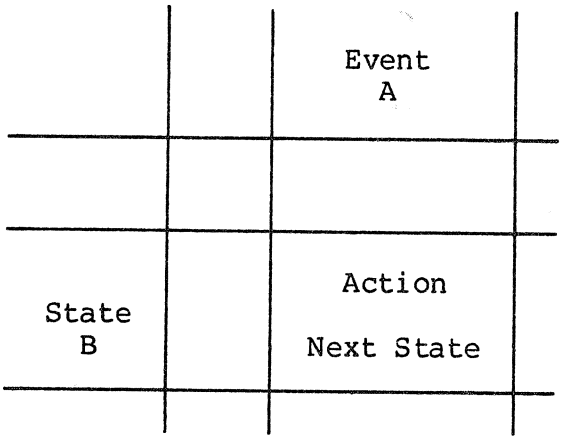
1. Introduction

The following tables show the likely action and state transition caused by each possible state-event pair. They present a possible implementation and, as such, seek to clarify the details of the protocol, but do not form a part of the specification. It is assumed here that all facilities can be used; it is fairly easy, however, to construct subsets of the tables which would be appropriate for any lesser combination of agreed facilities.

The tables are divided into ones for Level Ø and ones for Level 1. The Level Ø tables are further divided into P and Q, and Give and Take. The Level 1 tables are further divided into Sender and Receiver, and Normal conditions and Error and Termination conditions. This last division is somewhat arbitrary and is only necessary because a single table would be too large for the paper, because of the large number of possible events.

To save space the state, event and action names are abbreviated in the tables. However, each table is preceded by a summary of the states, events and actions that occur in it, showing the coding used for each.

Each entry in a table gives the action to be taken, at the top, and the state to be entered afterwards, at the bottom. 'No action' is indicated by a dash; two dashes indicate an internally generated event which should not happen in this state (such as 'End OK' when RR is outstanding). Thus each entry has the form:



The entry for event A occurring while in state B gives both the action to be taken and the state to be entered next.

The first state in eacn table is the starting state for that table. Level 1 is always entered via a state called 'GO', implying 'GO sent' or 'GO received' as appropriate. Level Ø can be re-entered via a number of states, depending on the way in which Level 1 is terminated. These states may at first appear to be redundant in the case of P, because the normal result is a spontaneous action of sending STOP and changing state accordingly. However, they are needed as distinct states to separate clearly Level Ø from Level 1; as a by-product this enables the exits from Level 1 to be shown symmetrically for both P and Q.

The following events take different shades of meaning depending on the state. 'Invalid command' includes the receipt of any command with an invalid parameter, or any command or data for which the entry is omitted or is given as an asterisk; at Level Ø it includes any Level 1 command. 'Time-out' means that the process has timed out awaiting either any input or a specific command. 'X error', where X can be P, Q, S (Sender) or R (Receiver), is either the first indication of a particular serious fault or, if an error has already occurred to get the process into its current state, then an indication that a more serious error has occurred.

The event 'TS Close' can occur at any time with obvious effect, so is not included in any of the tables; the tables are constructed so that Q normally performs the 'TS Close'.

2. Level Ø

2.1 Process P as Receiver

States:

SFT	SFT sent
ERok	ER(OK) sent
ERe	ER(E) sent
ABs	Abort, QR(A) sent
ABr	Abort, ES(A) received
STOP	STOP sent
GO	Enter Level 1 as receiver

Events:

None	spontaneous action
RPOS OK	acceptable
RPOS NOK	not acceptable
RNEG	rejected
Inv Com	invalid command
End	transfer not needed
TS reset	transport service reset
Timed out	a time-out has occurred
P error	error detected at P

Actions:

GO	enter data transfer phase
STOPt	STOP, indicating terminated
STOPr	STOP, indicating rejected
STOPa	STOP, indicating aborted
CLOSE	transport service close

2.2 Process P as Sender

The events and actions are the same as for Give above. The table is also the same, provided that some of the states are re-defined as follows:-

ERok	ER(OK) received
ERe	ER(E) received
ABs	Abort, ES(A) sent
ABr	Abort, QR(A) received
GO	Enter Level 1 as Sender

Event / State	None	RPOS OK	RPOS NOK	RNEG	Inv Com	End	TS reset	Timed out	P error
SFT	- SFT	GO GO	STOPr STOP	STOPr STOP	STOPa STOP	STOPt STOP	CLOSE -	STOPa STOP	STOPa STOP
ERok	STOPt STOP	*	*	*	- ERok	- -	CLOSE -	- -	STOPa STOP
ERe	STOPa STOP	*	*	*	- ERe	- -	CLOSE -	- -	CLOSE -
ABs	STOPa STOP	*	*	*	- ABs	- -	CLOSE -	- -	CLOSE -
ABr	STOPa STOP	*	*	*	- ABr	- -	CLOSE -	- -	CLOSE -
STOP	- STOP	*	*	*	- STOP	- -	CLOSE -	CLOSE -	CLOSE -

* Occurrence of this event in this state should be treated as an invalid command (Inv Com).

2.3 Process Q as Sender

States:

Open	transport service open
RPOS	RPOS sent
RNEG	RNEG sent
ERok	ER(OK) received
ERe	ER(E) received
ABs	Abort, ES(A) sent
ABr	Abort, QR(A) received

Events:

SFT OK	SFT judged acceptable
SFT NOK	SFT judged unacceptable
GO	GO received
STOP	STOP received
Inv Com	Invalid command received
TS reset	transport service reset
Timed out	a time-out has occurred
Q error	an error has been detected at Q

Actions:

RPOS	send a positive reply (viable)
RNEG	send a negative reply (rejected)
CLOSE	transport service close

2.4 Process Q as Receiver

The events and actions are the same as for Give above. The table is also the same, provided that some of the states are re-defined as follows:-

ERok	ER(OK) sent
ERe	ER(E) sent
ABs	Abort, QR(A) sent
ABr	Abort, ES(A) received
GO	Enter Level 1 as receiver

Event / State	SFT OK	SFT NOK	GO	STOP	Inv Com	TS reset	Timed out	Q error
Open	RPOS RPOS	RNEG RNEG	*	CLOSE -	CLOSE -	CLOSE -	CLOSE -	CLOSE -
RPOS	*	*	- GO	CLOSE -	CLOSE -	CLOSE -	CLOSE -	CLOSE -
RNEG	*	*	*	CLOSE -	CLOSE -	CLOSE -	CLOSE -	CLOSE -
ERok	*	*	*	CLOSE -	CLOSE -	CLOSE -	CLOSE -	CLOSE -
ERe	*	*	*	CLOSE -	CLOSE -	CLOSE -	CLOSE -	CLOSE -
ABs	*	*	*	CLOSE -	CLOSE -	CLOSE -	CLOSE -	CLOSE -
ABr	*	*	*	CLOSE -	CLOSE -	CLOSE -	CLOSE -	CLOSE -

* Occurrence of this event in this state should be treated as an invalid command (Inv Com).

3. Level 1

3.1 Normal Events for the Sender

States:

GO	GO sent or received
DATA	can send data
RR	RR received
HOLD	ES(H) sent
HORR	Hold state and RR received
MR	awaiting MR at acknowledgement window
ESok	ES(OK) sent
WAIT	waiting after transport service reset

Events:

MR	MR received
RR	RR received
ER(H)	ER(H) received
QR(OK)	QR(OK) received
QR(H)	QR(H) received
Data End	no more data to send
Mark Pnt	a mark insertion point has been reached
Rest Pnt	a point requested by an RR has been found
Ack W End	end of the acknowledgement window reached
Data OK	Data available for transmission
Set Code	Code change point or need code set after restart

Actions:

SS	send SS
CS	send CS
MS	send MS
ESok	send ES(OK)
ESh	send ES(H)
A-W	advance acknowledgement window limit
DATA	send data

Event / State	MR	RR	ER (H)	QR (OK)	QR (H)	Data End	Mark Pnt	Rest Pnt	Ack W End	Data OK	Set Code
GO	*	*	*	SS ESok ESok	SS ESh HOLD	SS ESok ESok	SS DATA	-	-	-	-
DATA	A-W DATA	- RR	*	ESok ESok	ESh HOLD	ESok ESok	MS A-W DATA	- -	- MR	DATA DATA	CS DATA
RR	A-W RR	*	*	*	ESh HORR	SS ESok ESok	- -	SS DATA	- -	- -	- -
HOLD	A-W HOLD	- HORR	- DATA	- HOLD	- HOLD	- -	- -	- -	- -	- HOLD	- -
HORR	A-W HORR	*	- RR	*	- HORR	- -	- -	- HORR	- -	- -	- -
MR	A-W DATA	- RR	*	ESok ESok	ESh HOLD	ESok ESok	- MR	- -	- -	- MR	- -
ESok	A-W ESok	- RR	*	- ESok	ESh HOLD	- ESok	- -	- -	- -	- -	- -
WAIT	A-W WAIT	- RR	*	ESok ESok	ESh HOLD	- WAIT	- -	- -	- -	- WAIT	- -

* Occurrence of this event in this state should be treated as an invalid command (Inv Com). The invalid command state is given in the next table, VI - 3.2.

3.2 Error and Termination Events for the Sender

States:

GO, DATA, RR, HOLD, HORR, MR, ESok, WAIT	As in 3.1 above
ESe	ES(E) sent
ERok	ER(OK) received - re-enter Level Ø
ERe	ER(E) received - re-enter Level Ø
ABs	ES(A) sent - re-enter Level Ø
ABr	QR(A) received - re-enter Level Ø

Events:

ER(OK)	ER(OK) received
ER(E)	ER(E) received
QR(E)	QR(E) received
QR(A)	QR(A) received
Inv Com	invalid command received
Timed out	a time-out has occurred
S error	local error detected by the sender
GO NOK	GO not acceptable
TS reset	transport service reset

Actions:

ESe	send ES(E)
ESa	send ES(A)

Event / State	ER (OK)	ER (E)	QR (E)	QR (A)	Inv Com	Timed out	S error	GO NOK	TS reset
GO	*	*	ESe ESe	- ABr	ESe ESe	- -	ESe ESe	ESe ESe	- WAIT
DATA	*	*	ESe ESe	- ABr	ESe ESe	- -	ESe ESe	- -	- WAIT
RR	*	*	ESe ESe	- ABr	ESe ESe	- -	ESe ESe	- -	- WAIT
HOLD	*	*	ESe ESe	- ABr	ESe ESe	ESa ABs	ESe ESe	- -	- WAIT
HORR	*	*	ESe ESe	- ABr	ESe ESe	ESa ABs	ESe ESe	- -	- WAIT
MR	*	*	ESe ESe	- ABr	ESe ESe	ESa ABs	ESe ESe	- -	- WAIT
ESok	- ERok	*	ESe ESe	- ABr	ESe ESe	ESa ABs	ESe ESe	- -	- WAIT
WAIT	*	*	ESe ESe	- ABr	ESe ESe	ESa ABs	ESe ESe	- -	- WAIT
ESe	- ERok	- ERe	- ESe	- ABr	- ESe	ESa ABs	- ESe	- -	ESe ESe

* Occurrence of this event in this state should be treated as an invalid command (Inv Com).

3.3 Normal Events for the Receiver

States:

GO	GO sent or received
DATA	ready to receive data
RR	RR outstanding (or initial SS awaited)
PEND	QR(H) outstanding
HOLD	ES(H) received in reply to QR(H)
PERR	RR state and QR(H) outstanding
HORR	RR state and ES(H) outstanding
ESok	ES(OK) received
QRok	QR(OK) sent
ERok	ER(OK) sent - re-enter Level Ø

Events:

SS	SS received
MS	MS received
CS	CS received
ES(OK)	ES(OK) received
ES(H)	ES(H) received
Data	Data received

OK for Ack	data sufficiently secure to be acknowledged
Hold up	need to enter Hold because of congestion
End hold up	congestion has cleared
End OK	enough of the file has been received
Data NOK	Data lost, need to re-transmit

Actions:

MR	send MR
RR	send RR
QRok	send QR(OK)
QRh	send QR(H)
ERok	send ER(OK)
ERh	send ER(H)
Save	record code or mark
Keep	store data

Event / State	SS	MS	CS	ES (OK)	ES (H)	Data	OK for Ack	Hold up	End hold up	End OK	Data NOK
GO	- DATA	*	*	*	*	*	- -	QRh PERR	- -	QRok QRok	- -
DATA	*	Save DATA	Save DATA	- ESok	- DATA	Keep DATA	MR DATA	QRh PEND	- -	QRok QRok	RR RR
RR	- DATA	- RR	- RR	- RR	- RR	- RR	MR RR	QRh PERR	- -	- -	- RR
PEND	*	Save PEND	Save PEND	- PEND	- HOLD	Keep PEND	MR PEND	- -	ERh DATA	ERh QRok QRok	RR PERR
HOLD	*	*	*	*	- HOLD	*	MR HOLD	- -	ERh DATA	ERh QRok QRok	RR HORR
PERR	- PEND	- PERR	- PERR	- PERR	- HORR	- PERR	MR PERR	- -	ERh RR	- -	- PERR
HORR	*	*	*	*	- HORR	*	MR HORR	- -	ERh RR	- -	- HORR
ESok	*	*	*	- ESok	- ESok	*	MR ESok	QRh PEND	- -	ERok ERok	RR RR
QRok	- QRok	- QRok	- QRok	ERok ERok	- QRok	- QRok	- QRok	- QRok	- QRok	- -	- -

* Occurrence of this event in this state should be treated as an invalid command (Inv Com). The invalid command state is given in the next table, VI - 3.4.

3.4 Error and Termination Actions for the Receiver

States:

GO, DATA, RR, PEND, HOLD, PERR, HORR, ESok, QRok	As in 3.3 above
ERok	
QRe	QR(E) sent
ERe	ER(E) sent - re-enter Level Ø
ABs	QR(A) sent - re-enter Level Ø
ABr	ES(A) received - re-enter Level Ø

Events:

ES(E)	ES(E) received
ES(A)	ES(A) received
Inv Com	invalid command received
Timed out	a time-out has occurred
R error	local error detected by the receiver
GO NOK	GO not acceptable to receiver
TS reset	transport service reset

Actions:

MR	send MR
RR	send RR
QRok	send QR(OK)
QRh	send QR(H)
QRe	send QR(E)
QRa	send QR(A)
ERe	send ER(E)

Event / State	ES (E)	ES (A)	Inv Com	Timed out	R error	GO NOK	TS reset
GO	ERe ERe	- ABr	QRe QRe	QRa ABs	QRe QRe	QRe QRe	RR RR
DATA	ERe ERe	- ABr	QRe QRe	QRa ABs	QRe QRe	- -	MR RR RR
RR	ERe ERe	- ABr	QRe QRe	QRa ABs	QRe QRe	- -	MR RR RR
PEND	ERe ERe	- ABr	QRe QRe	QRa ABs	QRe QRe	- -	MR RR RR
HOLD	ERe ERe	- ABr	QRe QRe	- -	QRe QRe	- -	MR QRh PEND
PERR	ERe ERe	- ABr	QRe QRe	QRa ABs	QRe QRe	- -	MR RR RR
HORR	ERe ERe	- ABr	QRe QRe	- -	QRe QRe	- -	MR RR RR
ESok	ERe ERe	- ABr	QRe QRe	- -	QRe QRe	- -	QRok QRok
QRok	ERe ERe	- ABr	QRe QRe	QRa ABs	QRe QRe	- -	QRok QRok
QRe	ERe ERe	- ABr	- QRe	QRa ABs	- QRe	- -	QRe QRe

Appendix VII

Members of the High Level Protocol Group

Mr. P.T. Barry
Computing Service,
James Watt (North) Bldg.
The University,
GLASGOW G12 8QQ

041-334 9771 Ext. 46

Mr. A.S. Chandler
Computer Aided Design Centre
Madingley Road,
CAMBRIDGE CB3 0HB

0223-63125

Dr. M.M. Curtis
Rutherford Laboratory,
Chilton,
Didcot,
OXON. OX11 0QX

0235-21900

Mr. P.M. Girard
Rutherford Laboratory,
Chilton,
Didcot,
OXON. OX11 0QX

0235-21900 Ext. 6331

Mr. M.J.T. Guy
Computer Laboratory,
Corn Exchange Street,
CAMBRIDGE CB2 3QG

0223-52435

Dr. K.S. Heard
Computer Science and Systems Division,
AERE,
Harwell,
Didcot,
OXON. OX11 0RA

0235-24141 Ext. 3022

Mr. P.L. Higginson
Department of Statistics and Computer Science,
University College London,
Gower Street,
LONDON WC1E 6BT

01-387 7050

Dr. P.F. Linington
Computer Laboratory,
Corn Exchange Street,
CAMBRIDGE CB2 3QG

0223-52435

Dr. M.A. McConachie
Cripps Computing Centre,
University of Nottingham,
NOTTINGHAM NG7 2HD

0602-56101 Ext. 3328

Dr. D. Rayner
National Physical Laboratory,
Teddington,
MIDDLESEX TW11 0LW

01-977 3222 Ext. 3996

Mr. N.R. Topham
Computer Aided Design Centre,
Madingley Road,
CAMBRIDGE CB3 0HB

0223-63125

Printed in Cambridge by the University Computing Service