

OFTP

Revision 1.4



Revision 1.4ABOUT THIS DOCUMENT

This new release (4) of OFTP includes, besides some error corrections, 2 major changes:

Addition of the Negative End Response (NERP).
Extended Date and Timestamp.

Odette WG4 has tried to incorporate the new items with as little effect as possible to the previous revisions.

The implementation of release 4 must follow the compatibility rules defined by downward negotiation at session set up as already stated in Rev 1.2.

SUMMARY OF CHANGES IN REVISION 1.4

The left margin change indications of Rev. 1.3 are eliminated.

Rev. 1.4 changes compared with Rev 1.3 are indicated in the left margin using the vertical bar character (|)

PREFACE REV 1.4 is added.

Value for Protocol Release Level clarified (3.4.28).

Addition of a command for Negative End Response (NERP) (1.6, 2.2, 2.4, 3.2, 3.3, 3.4, 6).

Extended Date and Timestamp (CCYYMMDD/HHMMSScccc) in the SFID and EERP commands (3.3, 3.4).

New reason code 14: File direction refused (3.4.15).

Corrections in the State Transition tables and diagrams (6).

OFTP	
Revision 1.4	PREFACE

Revision 1.3

ABOUT THIS DOCUMENT

This new release (2) of OFTP should be regarded as a clean-up release and includes, beside the addition of the clarification issues to the original document, only a minor correction of the ESID command and error recovery improvements to the special logic option.

In order to ease the implementation of special logic, a PAD parameter profile has been added.

Odette WG4 has tried to incorporate the new items with as little effect as possible to the original (rev 1.2) document.

The implementation of release 2 must follow the compatibility rules defined by downward negotiation at session set up as already stated in Rev 1.2.

SUMMARY OF CHANGES IN REVISION 1.3

The left margin change indications of Rev. 1.2 are eliminated.

Rev. 1.3 changes compared with Rev 1.2 are indicated in the left margin using the vertical bar character (|)

PREFACE REV 1.3 is added.

Preface header of rev 1.2 is added.

PROTOCOL VERSION LEVEL is changed to PROTOCOL RELEASE LEVEL in the SSID command (3.3 and 3.4.28)

A carriage return (C/R) is added to the ESID command (3.4).

An acronym of COMPLETE EXCHANGE BUFFER (CEB) is added (5.4).

Special logic synchronisation is clarified (5.4).

Addition of special logic error recovery (5.5).

Chapter number 5.5 - 5.8 changed to 5.6 - 5.9.

PAD parameter profile added (5.10)

CLARIFICATION ISSUES are added as an addendum.

Revision 1.2ABOUT THIS DOCUMENT

ODETTE group 4 have been charged with providing a communications facility for the electronic transmission of commercial data between trading partners within the motor manufacturing industry.

Group 4 selected the International Standards Organisation (ISO) Open System Interconnection (OSI) model as a basis for this communications facility.

CCITT X25 recommendation is the selected telecommunication protocol for OSI's layers 1, 2, 3.

The higher levels of OSI (levels 4 to 7) are still in the process of being defined and are expected to take many years before being totally defined. As a consequence, ODETTE group 4 have specified a file transfer protocol (FTP) which limits itself to addressing elementary file transfer. By defining a protocol that is so specific it is possible to have very short development timescales.

This document describes the FTP so agreed.

Exchanges of files composed of a single type of message have to be the functioning mode supported by all the ODETTE partners. Transfers of "composite files" (more than one type of message in a given file) being used only after bilateral agreement.

Any clearing centre has to be able to support the ODETTE specifications for file transfer.

In case of a link through a clearing centre, the functioning mode which has to be supported by the ODETTE partners must be the "simple transfer". Transfers with despatching and/or collation being used only after multilateral agreements.

SUMMARY OF CHANGES IN REVISION 1.2

Addition of a user field to the SFID and EERP commands.

Addition of a reserved field to the SFID, EERP and CDT commands.

Addition of seconds to the TIME parameter.

Addition of new argument value for the RECORD FORMAT parameter, in order to distinguish the "binary files" from the "text files".

Reduction of the size of the FILE NAME parameter.

VERSION LEVEL parameter placed immediately after the command code in the SSID command.

Correction of the state transition tables No 3 and No 4.

1. INTRODUCTION

- 1.1 - Objective
- 1.2 - General principles
- 1.3 - Structure
- 1.4 - Scope of FTP
- 1.5 - Network Service Requirements
- 1.6 - File Transfer Service
- 1.7 - X25 Addressing Restrictions

2. PROTOCOL DESCRIPTION

- 2.1 - Start Session phase
- 2.2 - Start File phase
- 2.3 - Data Transfer phase
- 2.4 - End of File phase
- 2.5 - End Session phase
- 2.6 - Problem Handling

3. COMMANDS AND FORMATS

- 3.1 - Conventions
- 3.2 - Commands
- 3.3 - Command layouts
- 3.4 - Parameters

4. DATA EXCHANGE BUFFER**5. SPECIAL LOGIC**

- 5.1 - When special logic is not to be used
- 5.2 - The need for "enveloping" exchange buffers
- 5.3 - Responsibilities of special logic
- 5.4 - Extended exchange buffer format
- 5.5 - Error recovery
- 5.6 - Sequence of events for special logic processing
- 5.7 - Checksum creation algorithm
- 5.8 - Algorithm for checking checksum parameters
- 5.9 - Shift-out processing
- 5.10 - PAD Parameter profile

6. STATE TRANSITION TABLES AND DIAGRAMS

- 6.1 - Input events
- 6.2 - Output events
- 6.3 - States
- 6.4 - Conventions for state tables
- 6.5 - Definition of local variables used by the FTP
- 6.6 - Definition of local constants used by the FTP
- 6.7 - State tables

7. ISO 646 CHARACTER SUBSET**ADDENDA**

- 1 - Clarification Issues
- 2 - Considerations for future development

I. Introduction

1.1 Objective

To define a protocol that allows transmission of a "character stream" (hereafter called file) in a non interactive way independent of the data communication network, regardless of system hardware, to and from one or more locations.

1.2 General principles

In the absence of an international standard for a FTP, this document will define a suitable protocol for use in the interim.

In designing and specifying the protocol, a number of considerations have been made:

- 1° The possible differences of size and sophistication.
(file storage, small and large systems)
- 2° The necessity to work with existing systems.
(reduce changes to existing products and allow easy implementation)
- 3° Systems of different ages.
- 4° Systems of different manufactures.
- 5° The potential for growth in sophistication.
(limit impact and avoid changes at other locations)

To allow with a minimum of bilateral technical agreement outside the FTP document, the data exchange between computer systems.

1.3 Structure

The description of a FTP is highly complex. To simplify the task, we have considered to use the level concept of the ISO reference model.

The protocol should be independent of the lower levels of the ISO reference model (level 1-3).

The FTP will run "directly" on the Connection Oriented Network Service (CONS, ISO IS 8348, CCITT rec. X213)

The FTP covers levels 4 to 7 and:

- 1° provides "File Service" to the monitor (local system environment),
- 2° makes use of the "Network Service".

ISO REFERENCE MODEL:

Level-7	FTP	application	← FILE SERVICE
Level-6	FTP	presentation	
Level-5	FTP	session	
Level-4	FTP	transport	← NETWORK SERVICE
Level-3		X.25	
Level-2		X.25	
Level-1		X.25	

1.4 Scope of FTP

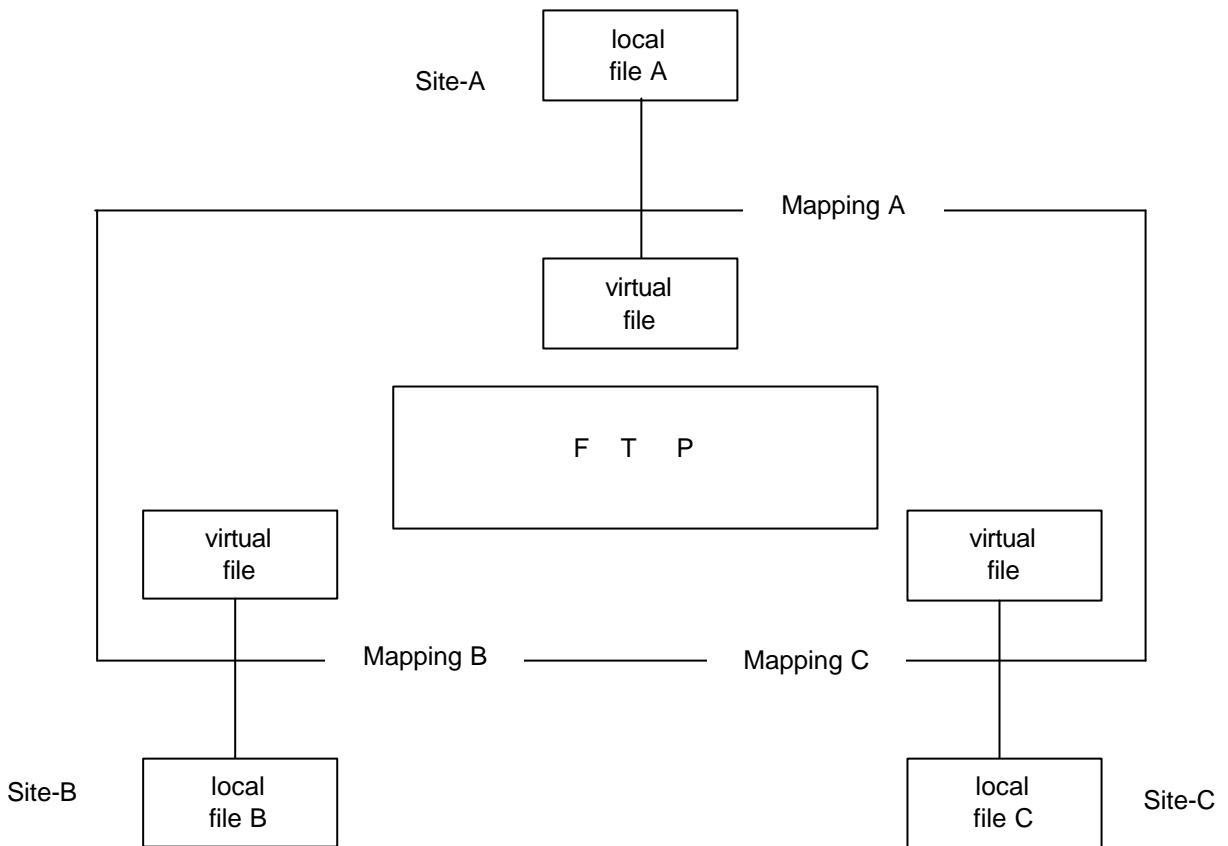
The FTP is built on the assumption that any file can be defined by a standard representation (virtual file).

The virtual file is described by a set of attributes identifying and defining the file to be transferred. The main attributes are:

- 1° FILE ORGANISATION: Sequential.
Logical records are presented one after another. The FTP must be aware of the records boundary.
- 2° FILENAME, DATE, TIME : to identify the transmitted data.
- 3° RECORD FORMAT:
 - . F (Fixed) : Each record in the file has the same length.
 - . V (Variable) : The records in the file can have a different length.
 - . U (Unstructured): The file contains a character stream of data. No predefined structure is available.
 - . T (Text File): A "Text File" is defined as a sequence of ASCII characters, containing no control character except CR/LF which delimitates lines. A line will not have more than 2048 characters. Restart works as for "U" files (i.e. using character count). If a given machine makes any transformation to the file, restart should work as if no transformation was performed (i.e. position is expressed as an offset from the beginning of the "virtual file").

The way in which a file is mapped from a local file to a virtual file, will vary from system to system and is part of the local system environment.

The exchange of information will always be in a standard representation.



1.5 Network Service Requirements

The network must be able to transport an integer number of octets (one NSDU : Network Service Data Unit).

No constraint may exist on NSDU content to inhibit this to allow transport of transparent data (no limitation in selection of character set).

The network must preserve the sequence and the boundaries of the NSDU's.

Exchange of NSDU's must be possible in half duplex mode (in one direction).

It must be possible to sent NSDU's with minimum length (see Section 4: data exchange buffer).

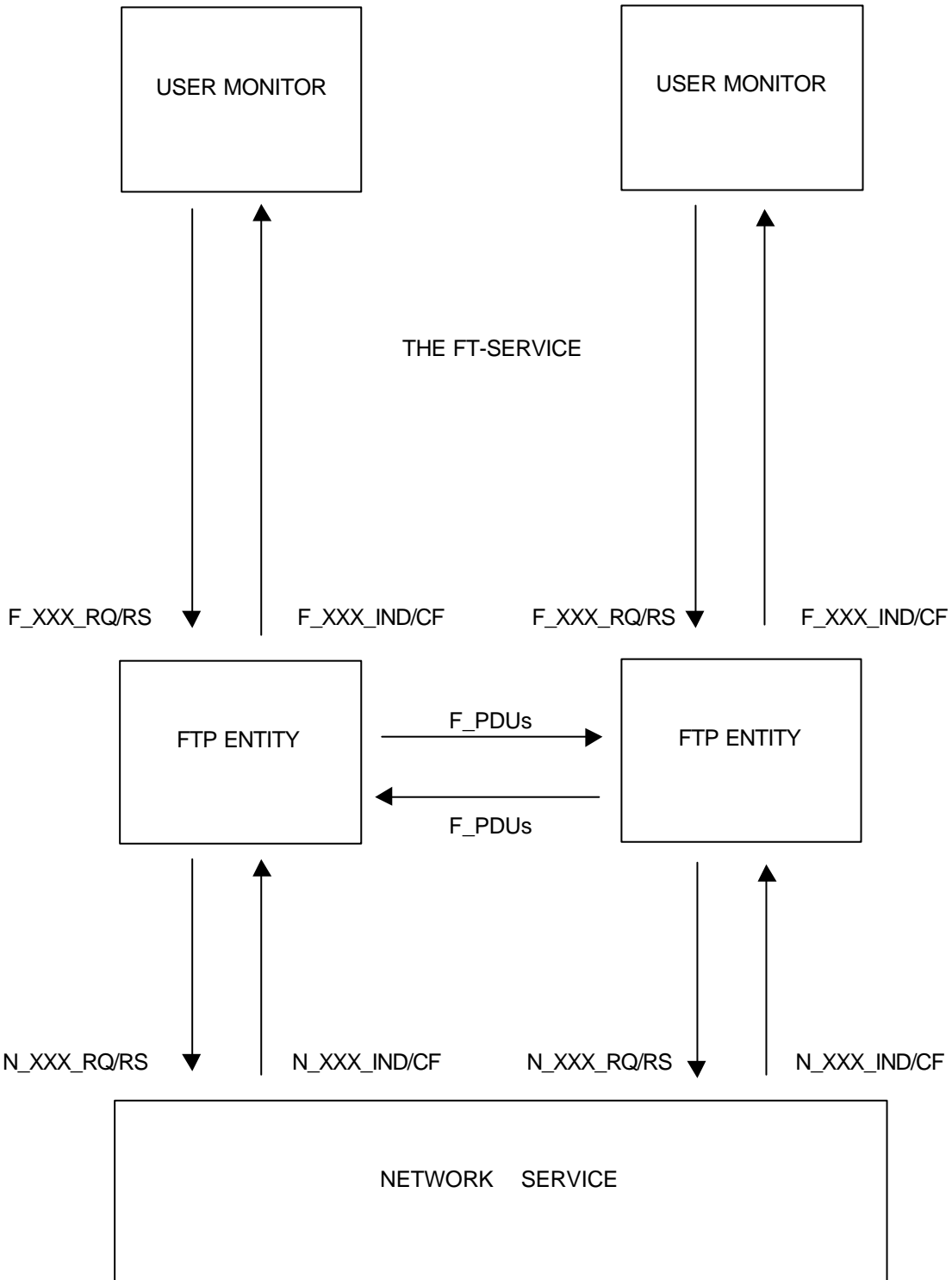
Must be able to handle contention situations.

The network must be able to pass length information of the NSDU to the FTP.

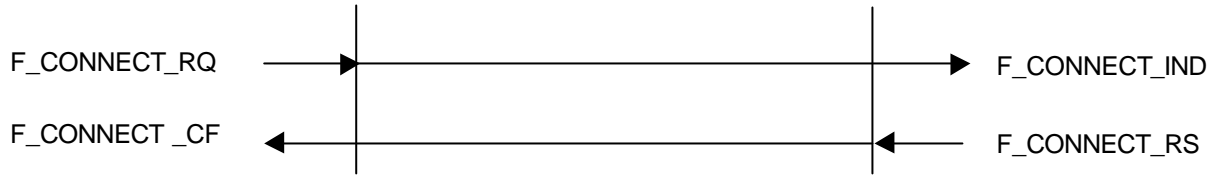
Service primitives: See Section 6.

1.6 File Transfer Service

1.6.1 Model



1.6.2 Session connection service



PARAMETERS:

Primitive	Request	Indication	Response	Confirm.
F_CONNECT	called-address	→ same	--	--
	calling-address	→ same	--	--
	ID1	→ same	ID2 →	→ same
	PSW1	→ same	PSW2 →	→ same
	model	→ mode2	→ mode3	→ same
	restart1	→ same	→ restart2	→ same

The negotiating rules for mode and restart are given below:

* MODE has 3 values:

- Sender-only: The entity which sends or receives a primitive with mode having this value, is only sender of files.
- Receiver-only: The entity which sends or receives a primitive with mode having this value, is only receiver of files.
- Both: Sender and receiver of files.

Negotiation between these values are:

- Both: Can be negotiated down to "Sender/Receiver".
- Sender/Receiver: Not negotiable.

F_CONNECT_RQ	F_CONNECT_IND	F_CONNECT_RS	F_CONNECT_CF
Sender-only →	Sender-only →	Sender-only →	Sender-only
Receiver-only →	Receiver-only →	Receiver-only →	Receiver-only
Both →	Both → or Receiver-only →	Both →	Both
		Receiver-only →	Sender-only
or or	Receiver-only →	Receiver-only →	Sender-only
	Sender-only →	Sender-only →	Receiver-only

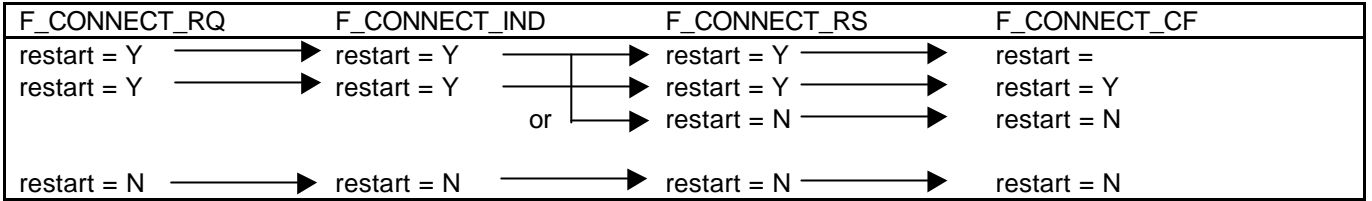
Notes:

Values in RS and CF are always the same; this is necessary in order to ensure that either entity (monitor) knows the same value at the end of the negotiation.

Both the acceptor (monitor) and the FT-Service (i.e. the 2 FTP entities) are entitled to perform the negotiation.

* RESTART has 2 values ("Y" and "N"), negotiation being performed from "Y" to "N".

The FT-Service is not allowed to negotiate the restart value.

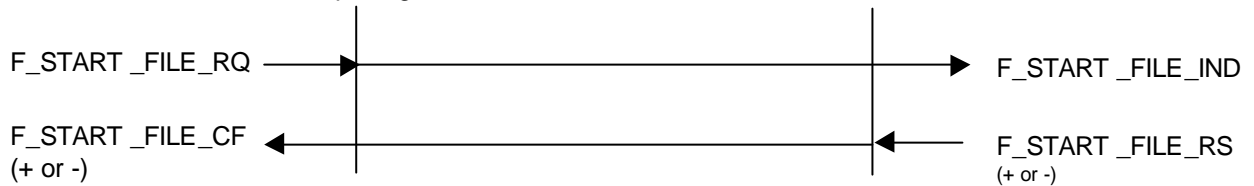


Notes:

Although present at the protocol level, compression capability, buffer-size, credit-value, special-logic, and carriage-return parameters are not seen at the service boundary, since they are managed internally to the FT-Service (i.e. by the 2 FTP entities).

1.6.3 File Transfer Phase

1.6.3.1 File Opening



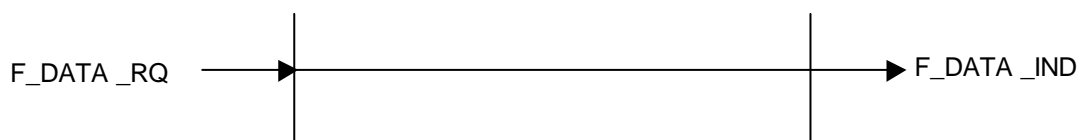
PARAMETERS:

Request	Indic.	Resp. (+)	Conf. (+)	Resp. (-)	Conf.(-)
file-name	same	--	--	--	--
date-time	same	--	--	--	--
destination	same	--	--	--	--
originator	same	--	--	--	--
rec-format	same	--	--	--	--
rec-size	same	--	--	--	--
file-size	same	--	--	--	--
restart-pos1	same	restart-pos2	same	--	--
--	--	--	--	cause	same
--	--	--	--	retry-later	same

Notes :

- Retry-later has values "Y" or "N".
- Cause is the reason for refusing to open the transfer (1, .. ,12).
- Restart-pos1 not equal 0 is only valid if restart has been agreed during initial negotiation.
- Restart-pos2 is less or equal than restart-pos1.

1.6.3.2 Data Regime

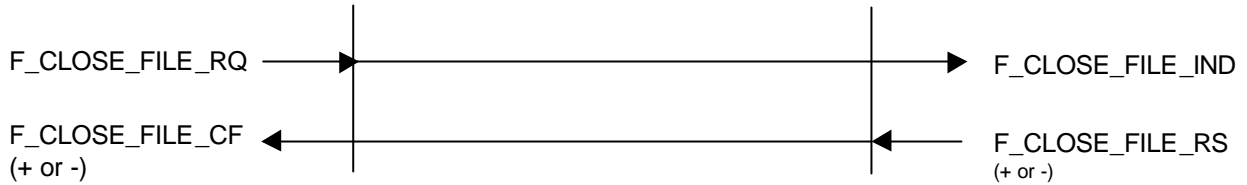


Notes :

The exact form in which data are exchanged in F_DATA primitives (i.e. record, records, part of record) is a local matter.

The FT-Service will also provide a flow control service which allows the receiving entity (monitor) to regulate the flow of F_DATA_IND: The FT-Service propagates this regulation to the sending entity.

1.6.3.3 File Closing



PARAMETERS:

Request	Indic.	Resp. (+)	Conf. (+)	Resp. (-)	Conf.(-)
rec-count	same	--	--	--	--
unit-count	same	--	--	--	--
--	--	speaker=Y	speaker=N	--	--
--	--	speaker=N	speaker=Y	--	--
--	--	--	--	cause	same

In F_CLOSE_FILE_RS the current speaker may either:

- Set speaker to "yes" in F_CLOSE_FILE_RS(+) and become speaker
- Set speaker to "no" in F_CLOSE_FILE_RS(+) and remain listener.

In all cases, the FT-Service set the speaker parameter of F_CLOSE_FILE_CF(+) to the value which indicates whether the current speaker remains speaker or becomes listener, in a consistent way with the capability of the peer user.

In case of negative RS/CF the turn is never changed, regardless of the value of the speaker parameter in the IND primitive.

Only the speaker is allowed to issue any of the F_XXX_FILE_RQ primitive.

1.6.4 Exchanging the Turn

1.6.4.1 Initial position of the turn (first speaker)

After the session open phase is completed (i.e. F_CONNECT_CF received by initiator and F_CONNECT_RS sent by responder), the initiator is always the first speaker.

1.6.4.2 Exchanging the turn

1° At each unsuccessful end of file the turn is not exchanged.

2° At each successful end of file the turn is changed if requested by the listener:

- The current listener receives F_CLOSE_FILE_IND (speaker = choice).
- If the listener answers F_CLOSE_FILE_RS (speaker = YES), it becomes speaker, the speaker receives F_CLOSE_FILE_CF (speaker = NO) and becomes listener.

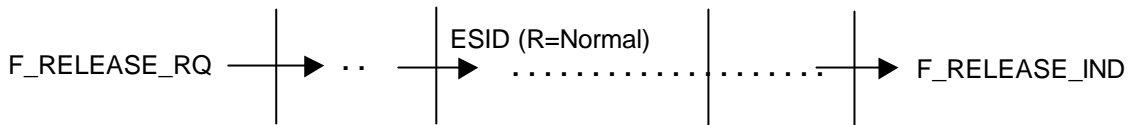
- If the listener answers F_CLOSE_FILE_RS (speaker = NO), it remains listener, and the speaker receives F_CLOSE_FILE_CF (speaker = YES) and remains speaker.

3° A speaker has the possibility to issues F_CD_RQ and becomes listener. The listener will receive F_CD_IND and becomes speaker.

4° In order to prevent loops of F_CD_RQ/IND, it is an error to send F_CD_RQ immediately after having received a F_CD_IND.

1.6.5 Session Closing Service

1.6.5.1 Normal closing



PARAMETERS:

Request	Indication
reason = normal	---

Rules for utilisation:

Only the current speaker can initiate the release service, provided there is no file in transfer.

Moreover, the speaker is only allowed to issue an F_RELEASE_RQ, just after receiving a F_CD_IND; this ensures that the other end is not willing to send more files.

Normal session release will be vehicled between two FTP's using the ESID command with : Reason = NORMAL release.

1.6.5.2 Abnormal closing

PARAMETERS:

Request	Indication
reason = error value	same (or equivalent) AO (Abort Origin): (L)ocal or (D)istant

Rules for utilisation:

Abnormal session release can be initiated by the speaker or the listener, also by the user or the provider.

Abnormal session release can occur at any time within the session.

Abnormal session release will be vehicled between two FTP's using the ESID command with : Reason = ERROR value.

The abnormal session release deals with the following types of error. Other error types may be treated by abort of connection.

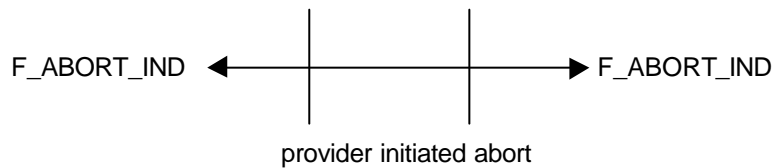
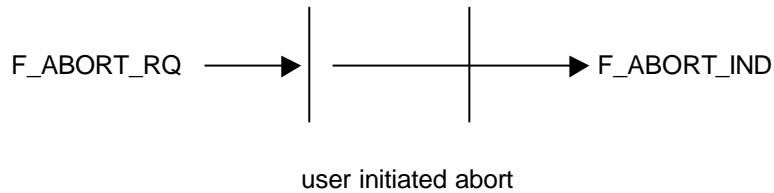
The service provider will initiate an abnormal release in the following cases:

- . Protocol error (violation),
- . Failure of SSID negotiation,
- . Command not recognized,
- . NSDU size error,
- . Resources not available,
- . Other unspecified abort code (with "REASON" = unspecified).

The user will initiate an abnormal release in the following cases:

- . Local site emergency close down,
- . Resources not available,
- . Other unspecified abort code (with "REASON" = unspecified).

1.6.6 Abort service

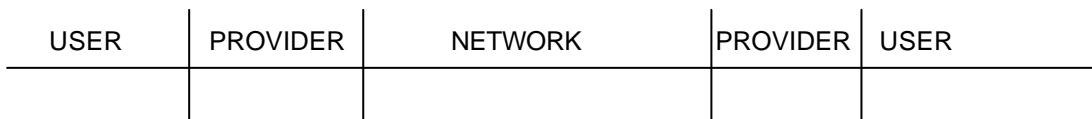


PARAMETERS:

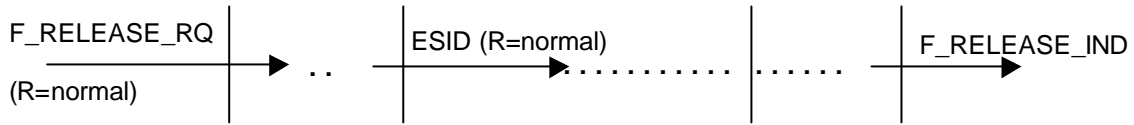
Request	Indication
--	R (Reason): specified or unspecified
--	AO (Abort Origin): (L)ocal or (D)istant

- At any time either entity (speaker or listener) may invoke the F_ABORT_RQ.
- The service provider may initiate abort in case of error detection.

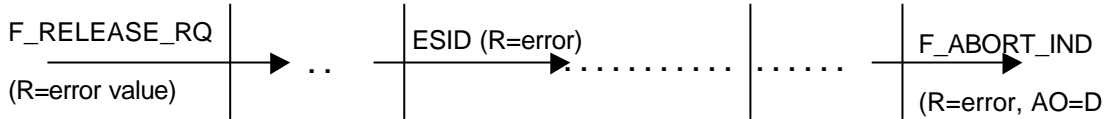
EXPLANATION OF NORMAL RELEASE, ABNORMAL RELEASE AND ABORT:



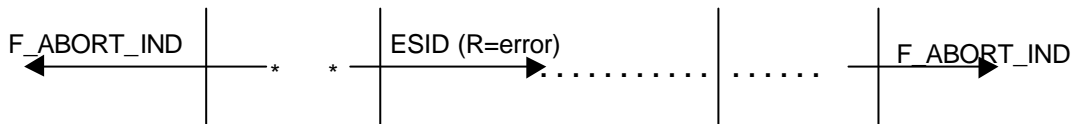
1. NORMAL RELEASE



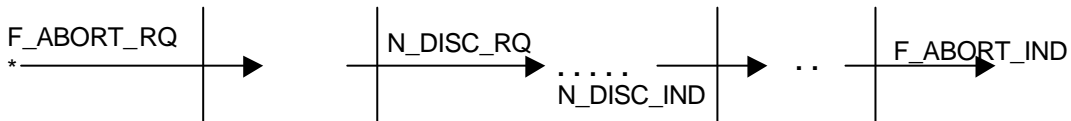
2. ABNORMAL SESSION RELEASE INITIATED BY THE USER



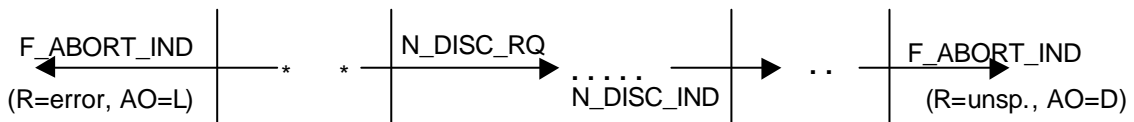
3. ABNORMAL SESSION RELEASE INITIATED BY THE PROVIDER



4. ABORT CONNEXION INITIATED BY THE USER



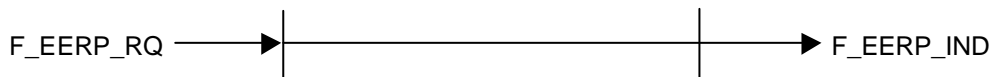
5. ABORT CONNEXION INITIATED BY THE PROVIDER



1.6.7 **End response service**

1.6.7.1 End-To-End Response

This service is initiated by the current speaker (if there is no file transfer in progress) to send an End-to-End response from the final destination to the originator of a file in case the file was successfully transmitted to the final destination.



PARAMETERS:

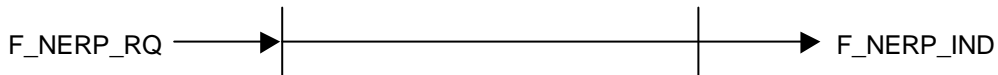
Request	Indication
filename	same
date	same
time	same
destination	same
originator	same

RELATIONSHIP WITH TURN:

- Only the current speaker (the turn owner) can send F_EERP_RQ.
- Invoking the EERP service does not change the turn.
- If a F_CD_IND has been received just before F_EERP_RQ is issued, this results in leaving the special condition created by the reception of F_CD_IND; i.e. while it was possible to issue F_RELEASE_RQ and not possible to issue F_CD_RQ just after the reception of F_CD_IND, after having issued F_EERP_RQ the normal speaker status is entered again (F_CD_RQ valid, but F_RELEASE_RQ not valid).

1.6.7.2 Negative End Response

This service is initiated by the current speaker (if there is no file transfer in progress) to send a negative end response in case a file could not be transmitted to the next destination. It will only be send if the problem is of a non temporary kind.



PARAMETERS:

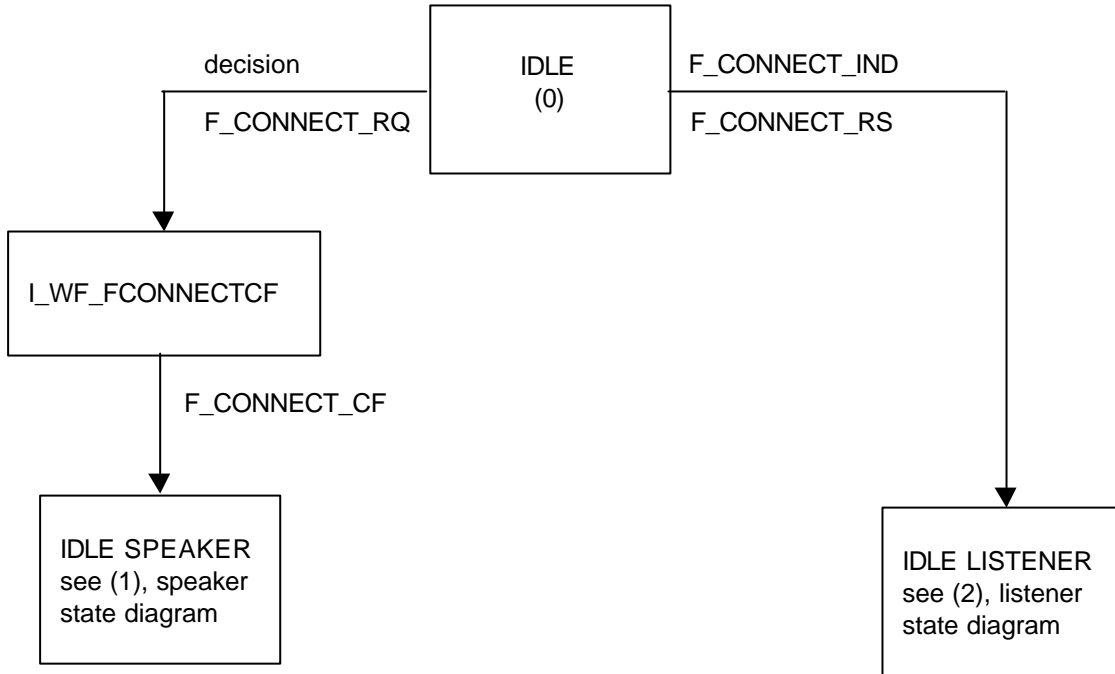
Request	Indication
filename	same
date	same
time	same
destination	same
originator	same
creator of negative response	same
reason	same

RELATIONSHIP WITH TURN:

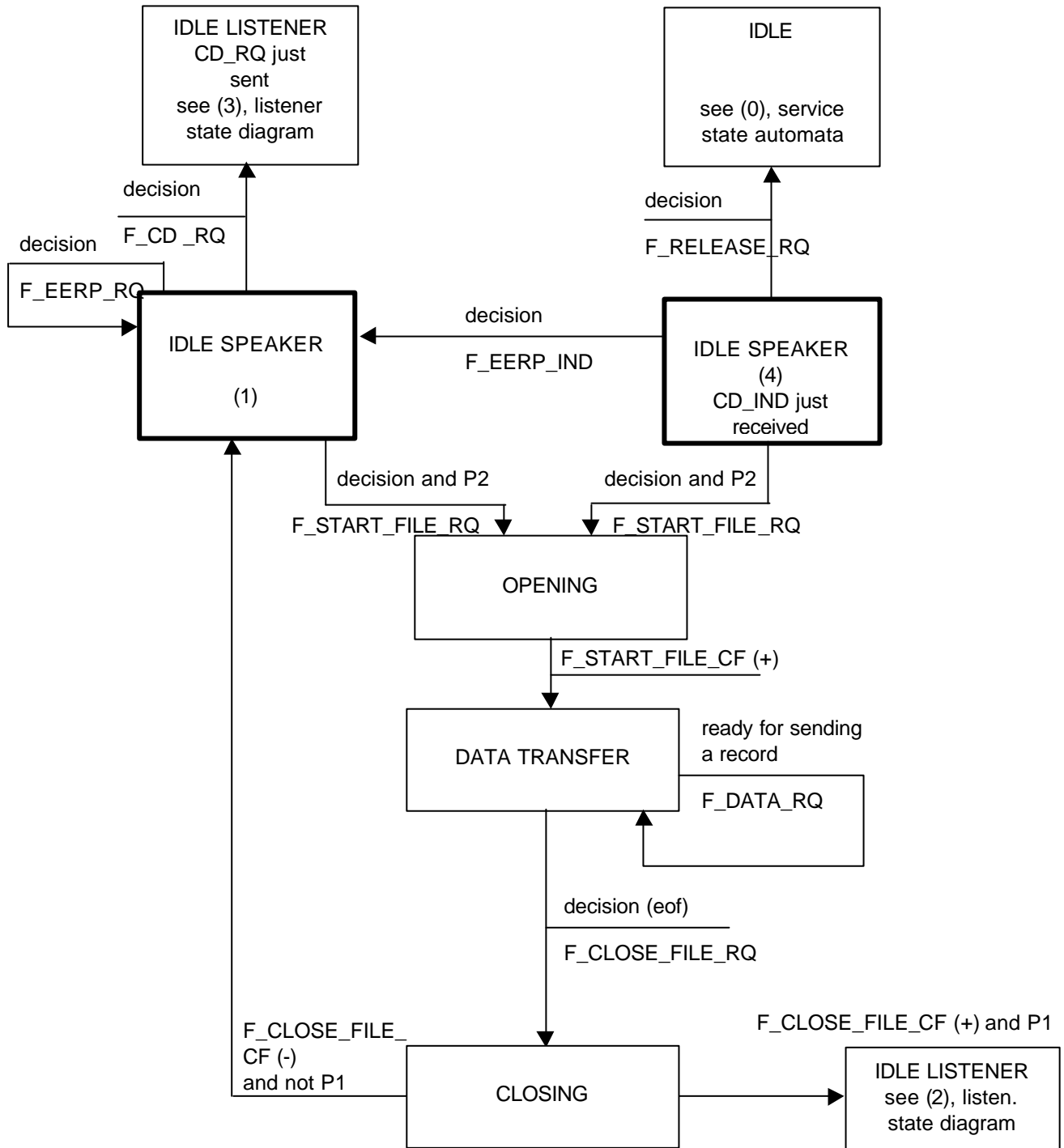
The same as for the End-To-End response (see 1.6.7.1).

1.6.8 Service state automata

This state automata defines the service as viewed by the user.



1.6.9 Speaker state diagram

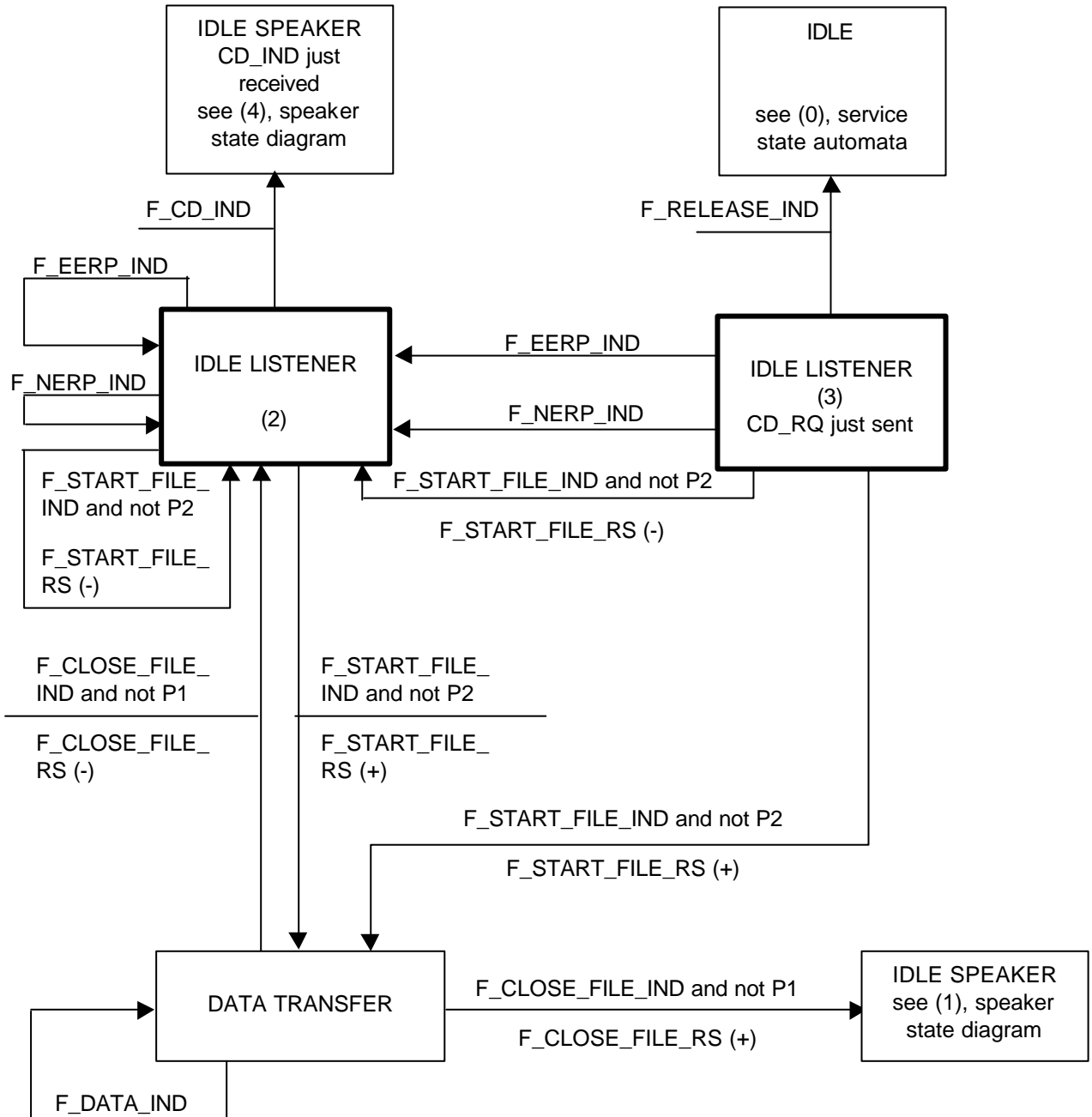


ADVANCE ADVANCE ADVANCE ADVANCE ADVANCE ADVANCE ADVANCE ADVANCE ADVANCE

P1 : Negative confirmation or (positive confirmation and speaker = yes)

P2 : Mode = both or (mode = sender-only)

1.6.10 Listener state diagram



ADVANCEADVANCEADVANCEADVANCEADVANCEADVANCEADVANCEADVANCEADVANCEADVANCE

P1 : (decision to send F_CLOSE_FILE_RS(+)) AND
 (decision to set speaker = yes in F_CLOSE_FILE_RS(+))
 P2 : (decision to send F_START_FILE_RS(+))

1.7 X.25 Addressing Restrictions

When an X25 call is made over a PSDN, the NUA of the destination must be specified in order that the PTT may route the call. The call placed is directed to the termination equipment upon the user's premises.

It is possible to provide extra information in the Call Request Packet in addition to the mandatory NUA required by the PTT.

This extra information may be of 2 kinds:

a) A sub-address:

It is simply an extension to the address and it is put into the called address field of the Call Request Packet. This information (Address + Sub-address) is taken from the destination address field of the F_CONNECT_RQ, therefore from the user's point of view there is no distinction between the part which is the main address and the part which is the sub-address.

b) User data:

There is no standard for user data. Moreover there is no information in the F_CONNECT_RQ from which the Odette-entity may derive user data to be put in the N_CONNECT_RQ; therefore User data shall not be used.

II. Protocol description

2.1 Start Session phase

ENTITY DEFINITION:

The Start Session phase starts after the network connection.

The entity that took initiative to establish the network connection becomes the INITIATOR. The other is called the RESPONDER. If the network service cannot detect the entity that took the initiative, an "initiator/responder" entity will be predefined.

PROTOCOL SEQUENCE:

The first message must be sent by the RESPONDER.

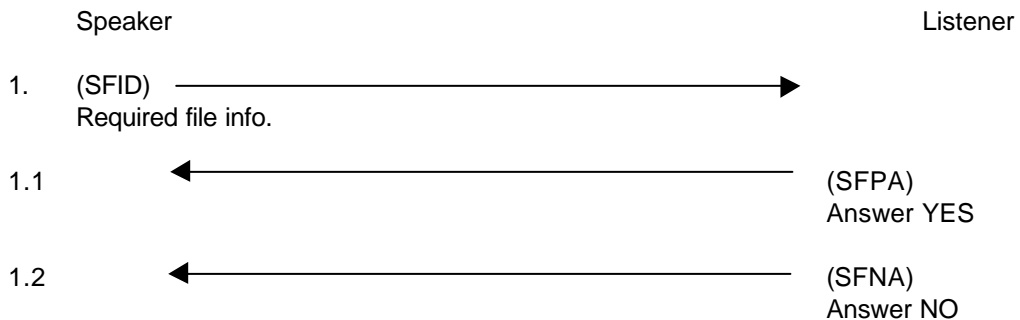


2.2 Start File phase

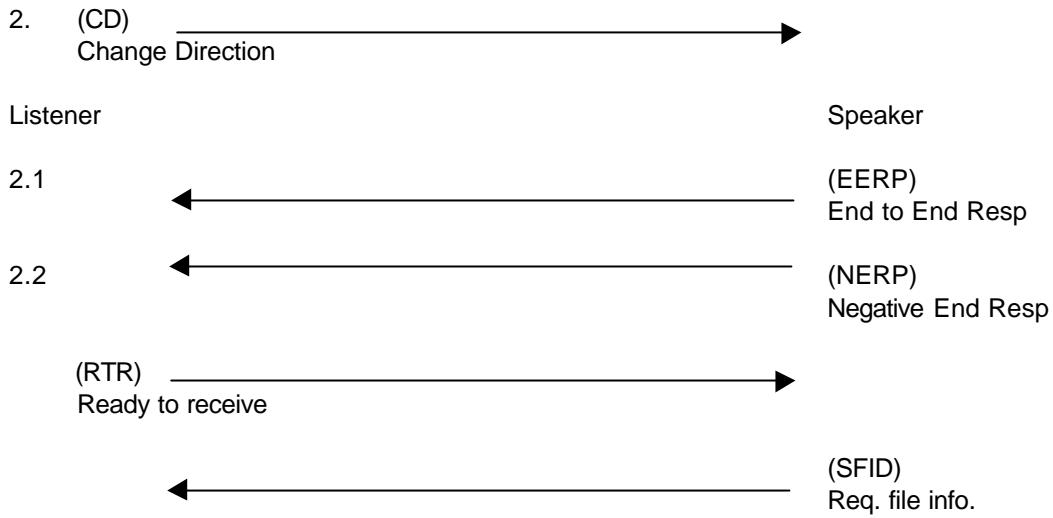
ENTITY DEFINITION:

The entity of SPEAKER or LISTENER is the result of the Start Session phase, where the INITIATOR becomes the first SPEAKER or as a result of a change direction request.

PROTOCOL SEQUENCE:



GO TO 1. The monitor should prevent a loop situation, or prevent retransmission.



RESTART FACILITIES:

The required fields will include a record count. If no restart facilities are available, a record count of zero should be passed. The sender will start with the lowest record count + 1.

BROADCAST FACILITIES:

The destination field of the required file information can be:

- An explicit defined destination,
- A group destination that allows distribution (based on tables) at an intermediate location of the virtual file to several destinations.

The listener will send a negative answer to the speaker when the destination is not known.

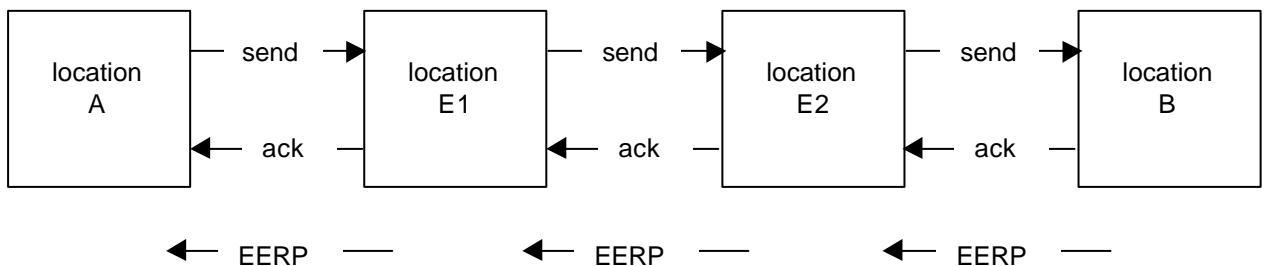
PRIORITY:

It is left to the local implementation to allow priorities for the files it has to send.

To allow some flexibility, a change direction has been implemented in the End of File phase (see hereafter).

END TO END CONTROL:

To allow end to end control, a RESPONSE COMMAND has been created.



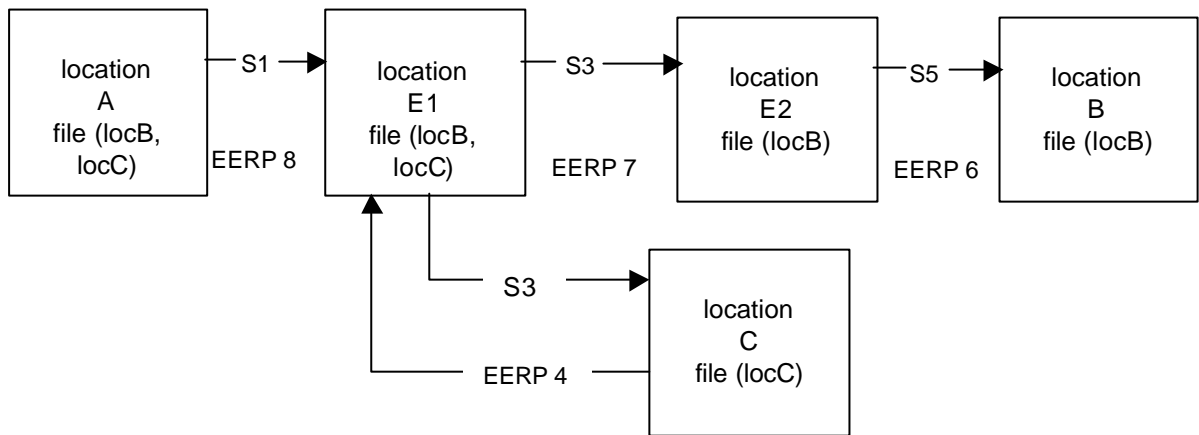
The response command will be created by the location that performs process of the data, distribution or collection of the data. It must send the EERP to the originator to indicate that the file has been received. The EERP command is mandatory in the protocol.

The response command is a notification to the originator of the file, that the data was delivered in good manner. This will allow the sender to do house keeping functions (e.g. delete of delivered files).

It is obvious that depending on the functions performed, an intermediate node (clearing centre) must keep track of the origin and destination of files and messages (see examples hereafter).

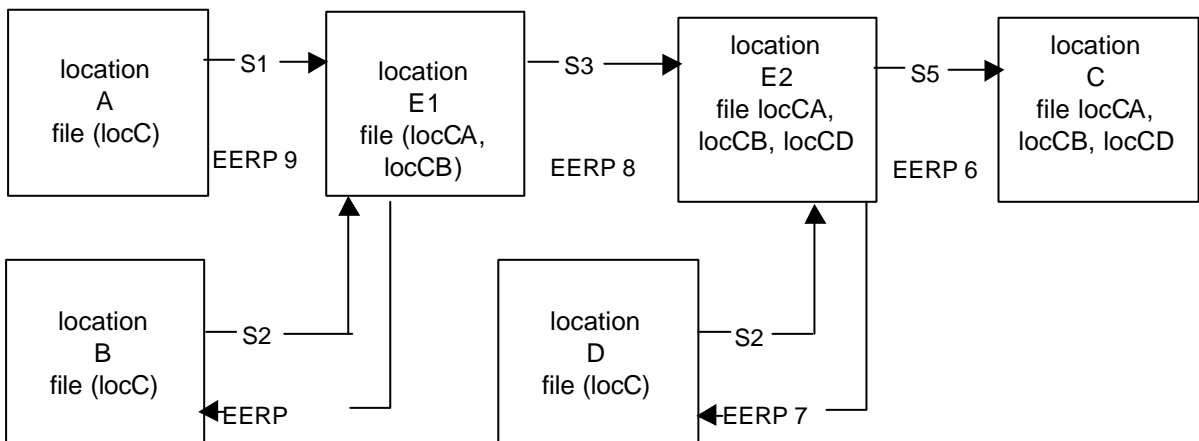
The EERP contains parameters (see section 4) that are identical to the SFID but will be interpreted by the local system environment "monitors" quite differently. This command flows back from Final Destination to Originator.

Example: Data distribution



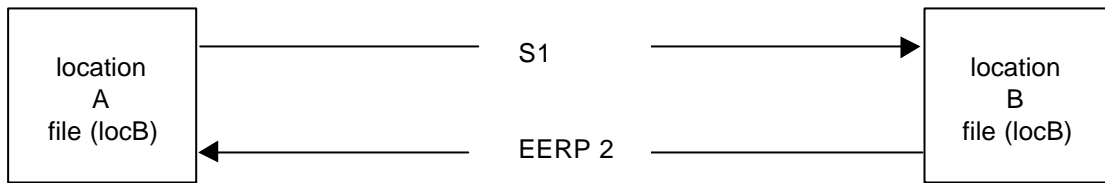
Note: EERP-8 waits for End-to-End Response of location E2 (for B) and location C.

Example: Data collection



Note: Location E1 can only send confirmation to location A and B after receipt of End-to-End Response of location E2.

Example: Point to point



Note: Location B will send EERP to location A after receipt of complete file.

NEGATIVE END CONTROL

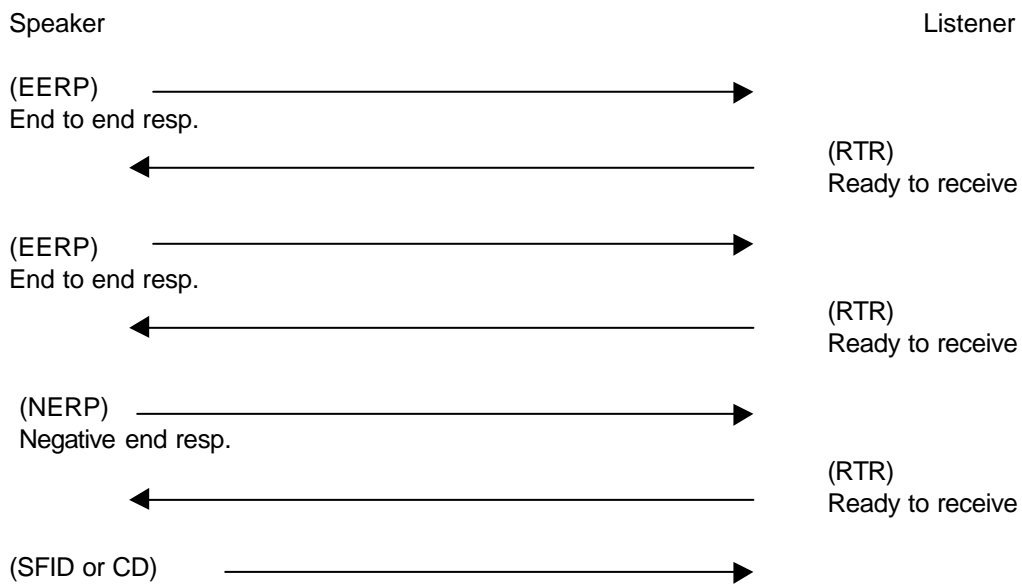
In addition to the EERP which allows control over successful transmission of a file, a Negative End Response signalizes that a file could not be delivered to the final destination. It will be created by an intermediate node that could not transmit the file any further, because the next node refuses to accept the file. The cause of the refusal has to be of a non-temporary kind, otherwise the intermediate node has to try the transmission again.

The NERP will be sent back to the originator of the file. The parameters are equal to the ones of the EERP, but additional with information about the creator of the NERP and the abort reason. The abort reason is taken from the refusal reason that was send by the node denying the file.

Because of the NERP it is possible for the intermediate node to stop trying to send the not-deliverable file and to delete it. And for the originator of the file it will be possible to react on the unsuccessful transmission, depending on the reason code and the creator of the NERP.

READY TO RECEIVE COMMAND: RTR

In order to avoid congestion between 2 adjacent nodes due to a possible non interrupted flow of EERP's and NERP's, an RTR command (Ready to receive) is provided. RTR is then the acknowledgement to EERP and NERP but has no end-to-end significance.



After sending an EERP or NERP, the sender will wait for an RTR before sending any other commands:

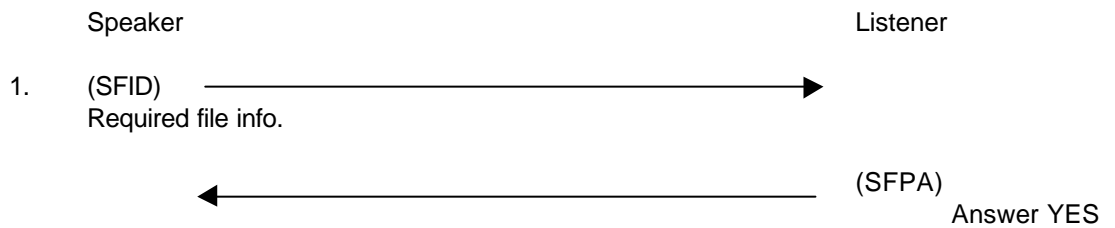
- . EERP, NERP
- . SFID or CD if there is no more EERP or NERP to be sent.

2.3 Data Transfer phase

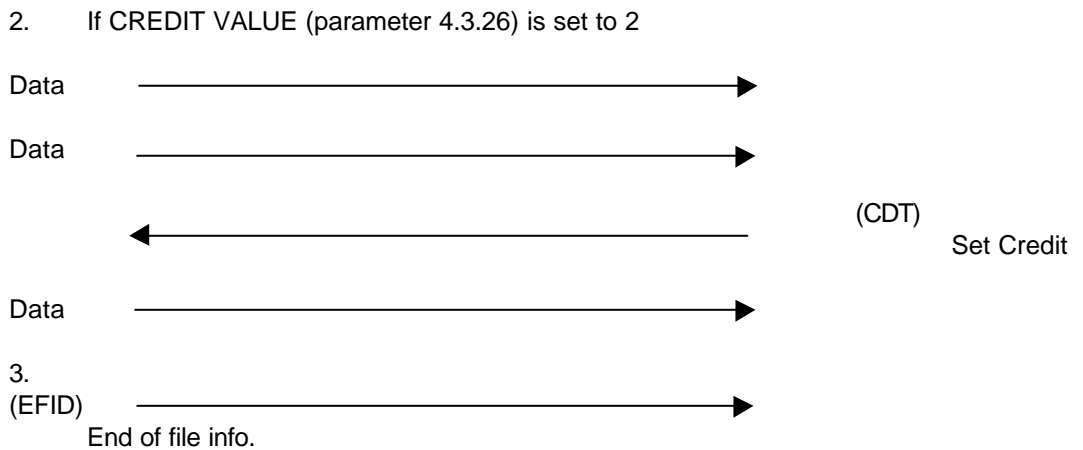
PROTOCOL SEQUENCE:

After the Start File phase, data will flow from speaker (sender) to listener (receiver).

To avoid congestion at the FTP level a flow control mechanism is provided via CDT. The speaker has not the right to send data unless he has the permission of the listener. The number of Data Exchange Buffers that the speaker is allowed to send is negotiated in the Start Session phase. Sending more data than allowed will result in protocol error and leads to an abort. The listener should not delay sending the CDT in order to avoid speaker blocking.

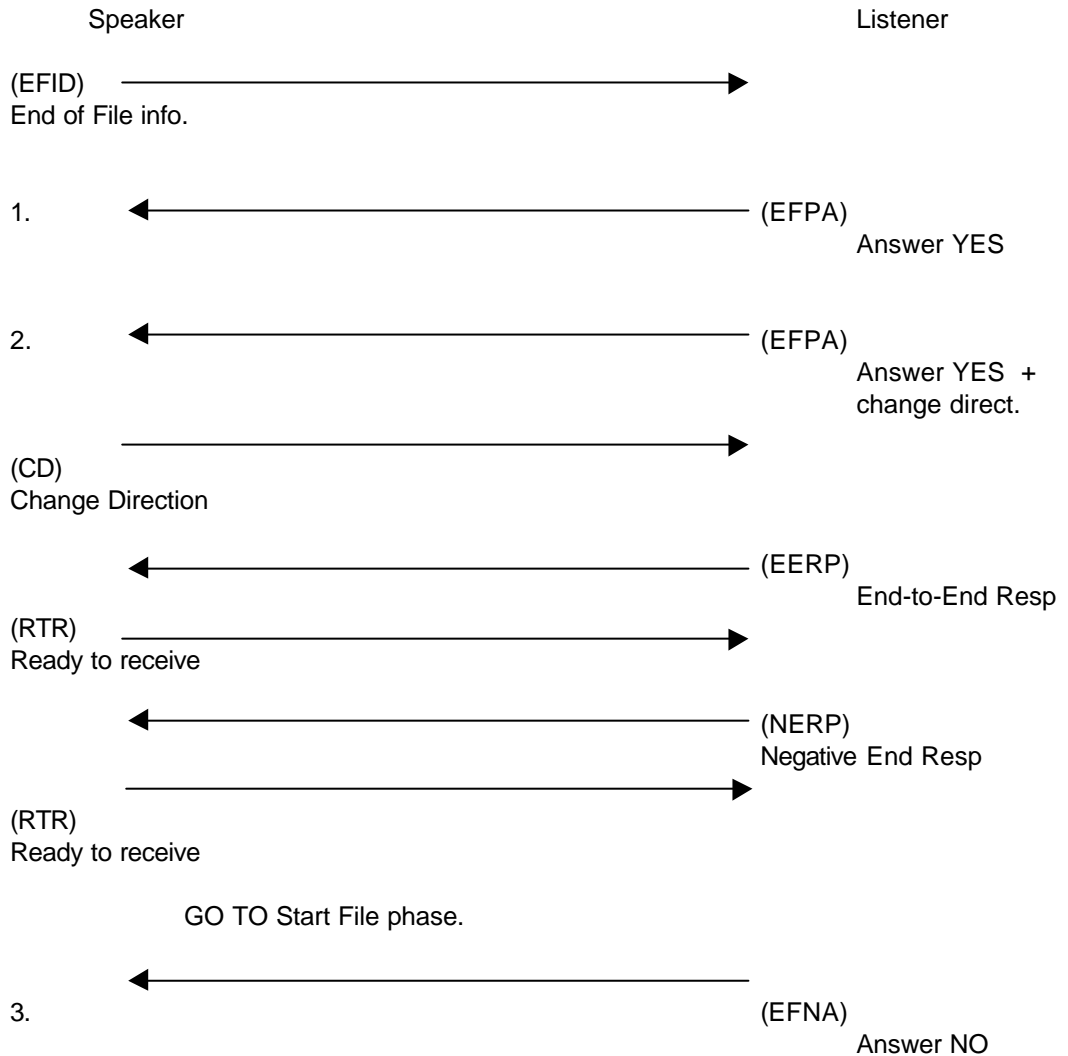


The SFPA indicates that speaker is allowed to send data. It is an implicit Set Credit.



2.4 End of File phase

PROTOCOL SEQUENCE:

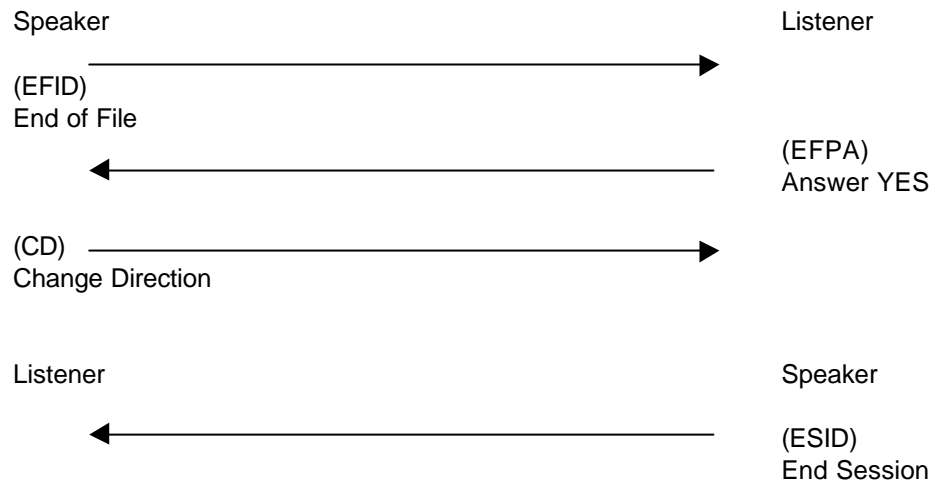


The monitor at the sender site must initiate a retransmission.

2.5 End Session phase

PROTOCOL SEQUENCE:

The last sender (speaker) will take the initiative to terminate the session.

**2.6 Problem Handling**

As a base principle, error detection and error handling should be done as close as possible to the problem medium, this is to:

- 1° Clearly identify the problem.
- 2° Use the correct tools to correct the problem.

Within the ISO reference model, each layer is responsible for his functional error handling. The FTP will provide an error detection and problem resolving protocol.

We should not mix end-to-end control and error detection/problem solving.

When using clearing centres, a probability exists that errors can occur due to file manipulation in the clearing centre. These errors cannot be related to the FTP or communication levels, but are the result of data processing activities.

RULES:

- 1° If a protocol error occurs in one of the above mentioned phases, the session will terminated and application activity aborted.
- 2° When we work on a switched link, if an error occurs, the link will be disconnected.

When working on a non-switched link, both locations enter the IDLE status.

TIMEOUT: (see Section 6 for TIMERS)

Is the event which occurs after a defined period of time and which assists with the detection of error conditions. It is used to time all situations where a response is required. It allows for unilateral termination of the transmission. It prevents hang situations.

- 1° If timeout situations occur, actions will be taken as described in the rules.
- 2° As default value, 10 minutes has been chosen. In any case, the selected value should take in account all acceptable network delays, in both directions.

III. Commands and formats

3.1 Conventions

REPRESENTATION UNIT:

The octet (eight bits), is considered as a unit by the FTP.

VALUES AND CHARACTERS:

The ISO 646 IRV 7-bits coded character set for information processing exchange has been selected.

3.2 Commands

Each command has a fixed length and cannot be compressed. There is no maximum length for all commands.

Commands and data are not mixed in the DATA EXCHANGE BUFFER. A command start at the beginning of the buffer.

COMPONENTS:

- 1° Command identifier:
The command identifier is a single octet (see hereafter).
- 2° Parameter(s):
There may be as many parameters as needed, but:
 - predefined order (sequence as they are specified in the TABLE hereafter)
 - positional
 - required (no default value).

TABLE:

COMMAND (char)	NAME	DESCRIPTION	PHASE	USED BY ENTITY	PARAMETERS
I	SSRM	ready message	2.1	- R	3.4.22/3.4.23
X	SSID	identification, password and profile	2.1	I - R	3.4.28/3.4.01 3.4.02/3.4.03 3.4.04/3.4.05 3.4.20/3.4.24 3.4.26/3.4.21 3.4.29/3.4.23
H	SFID	file information	2.2	S -	3.4.06/3.4.30 3.4.07/3.4.08 3.4.31/3.4.09 3.4.10/3.4.25 3.4.11/3.4.12 3.4.13
2	SFPA	positive answer	2.2	- L	3.4.14
3	SFNA	negative answer	2.2	- L	3.4.15/3.4.16
D		data	2.3		section 4 section 5
C	CDT	set credit	2.3	- L	3.4.32
T	EFID	end of file information	2.4	S -	3.4.17/3.4.18
4	EFPA	positive answer	2.4	- L	3.4.19
5	EFNA	negative answer	2.4	- L	3.4.15
F	ESID	end of session	2.5	S -	3.4.27
R	CD	change direction (this command gives the right to talk)	2.2 2.4 2.5	S - S - I - R	none
E	EERP	end-to-end response command	2.2 2.4	S -	3.4.06/3.4.30 3.4.07/3.4.08 3.4.31/3.4.09 3.4.10
E	NERP	negative end response command	2.2 2.4	S -	3.4.06/3.4.32 3.4.07/3.4.08 3.4.09/3.4.10 3.4.33/3.4.34
P	RTR	ready-to-receive command	2.2 2.4	- R	none

3.3 Command layouts

The following record layouts are described using five columns per field entry.

FIELD:

- 01 Field name.
- 02 Contains either an 'F' or a 'V'. The 'F' describes a field with a fixed or finite number of values e.g. SSIDREST can only have the values 'Y' or 'N'. The 'V' describes a field which is totally variable within the confines of it's data type e.g. SFIDREST can have many values in the range 000000000 to 999999999.
- 03 The field type. The 'X' defines an alphanumeric field and a '9' describes a numeric field. The bracketed number following gives the field width in bytes. Therefore X(25) defines an alphanumeric field of 25 octets.
- 04 General field description.
- 05 Field reference number. This reference number may be used to locate a detailed description under the section 3.4 "Parameters" which follows the record layouts.

	SSRM		READY MESSAGE	
SSIDCMD	F	X(1)	COMMAND 'I'	
SSRMMSG	F	X(17)	'ODETTE FTP READY'	22
SSRMCR	F	X(1)	CARRIAGE RETURN	23

	SSID		IDENTIFICATION PASSWORD & PROFILE	
SSIDCMD	F	X(1)	COMMAND 'X'	
SSIDLEV	F	9 (1)	PROTOCOL RELEASE LEVEL	28
SSIDCODE	V	X(25)	INITIATOR'S ID CODE	01
SSIDPASWD	V	X(8)	INITIATOR'S PASSWORD	02
SSIDSDEB	V	9(5)	EXCHANGE BUFFER SIZE	03
SSIDSR	F	X(1)	(S)END (R)ECEIVE (B)OTH	04
SSIDCMPR	F	X(1)	COMPRESSION IND (Y/N)	05
SSIDREST	F	X(1)	RESTART OPTION (Y/N)	20
SSIDSPEC	F	X(1)	SPECIAL LOGIC IND (Y/N)	24
SSIDCRED	V	9(3)	EXCHANGE BUFFER CREDITS	26
SSIDSV1	F	X(5)	RESERVED	21
SSIDUSER	V	X(8)	FIELD AVAILABLE TO USER	29
SSIDCR	F	X(1)	CARRIAGE RETURN	23

OFTP

Revision 1.4

COMMANDS AND FORMATS

SFID		SEND FILE INFORMATION		
SFIDCMD	F	X(1)	COMMAND 'H'	
SFIDDSN	V	X(26)	FILE DATASET NAME	06
SFIDRSV1	F	X(3)	RESERVED	30
SFIDDATE	V	9(8)	DATE CCYYMMDD	07
SFIDTIME	V	9(10)	TIME HHMMSScccc	08
SFIDUSER	V	X(8)	FIELD AVAILABLE TO USER	31
SFIDDEST	V	X(25)	DESTINATION	09
SFIDORIG	V	X(25)	ORIGINATOR	10
SFIDFMT	F	X(1)	FILE FORMAT F/V/U/T	25
SFIDLRECL	V	9(5)	MAXIMUM RECORD SIZE	11
SFIDFSIZ	V	9(7)	FILE SIZE IN 1K BLOCKS	12
SFIDREST	V	9(9)	RESTART POSITION	13

SFPA		POSITIVE ANSWER		
SFPACMD	F	X(1)	COMMAND '3'	
SFPAACNT	V	9(9)	ANSWER COUNT	14

SFNA		NEGATIVE ANSWER		
SFNACMD	F	X(1)	COMMAND '3'	
SFNAREAS	F	9(2)	ANSWER REASON	15
SFNARRTR	F	X(1)	ANSWER RETRY (Y/N)	16

CDT		SET CREDIT		
CDTCMD	F	X(1)	COMMAND 'C'	
CDTRSV1	F	X(2)	RESERVED	32

EFID		END OF FILE INFORMATION		
EFIDCMD	F	X(1)	COMMAND 'T'	
EFIDRCNT	V	9(9)	RECORD COUNT	19
EFIDUCNT	V	9(12)	UNIT COUNT	

EFPA		END OF FILE POSITIVE ANSWER		
EFPACMD	F	X(1)	COMMAND '4'	
EFPACD	F	X(1)	CHANGE DIRECTION (Y/N)	19

EFNA		END OF FILE NEGATIVE ANSWER		
EFNACMD	F	X(1)	COMMAND '5'	
EFNAREAS	F	9(2)	ANSWER REASON	15

ESID		END OF SESSION		
ESIDCMD	F	X(1)	COMMAND 'F'	
ESIDCMD	F	9(2)	REASON CODE	27
ESIDCR	F	X(1)	CARRIAGE RETURN	23

OFTP

Revision 1.4

COMMANDS AND FORMATS

CD		CHANGE DIRECTION	
CDCMD	F	X(1)	COMMAND 'R'

EERP		READY TO RECEIVE		
EERPCMD	F	X(1)	COMMAND 'E'	
EERPDSN	V	X (26)	FILE DATASET NAME	06
EERPRSV1	F	X (3)	RESERVED	30
EERPDATE	V	9 (8)	DATE CCYYMMDD	07
EERP TIME	V	9 (10)	TIME HHMMSScccc	08
EERPUSER	V	X (8)	FIELD AVAILABLE TO USER	31
NERPDEST	V	X (25)	DESTINATION	09
NERPORIG	V	X (25)	ORIGINATOR	10

NERP		NEGATIVE END RESPONSE COMMAND		
NERPCMD	F	X (1)	COMMAND 'N'	
NERPDSN	V	X (26)	FILE DATASET NAME	06
NERPRSV1	F	X (6)	RESERVED	33
NERPDATE	V	9 (8)	DATE CCYYMMDD	07
NERP TIME	V	9 (10)	TIME HHMMSScccc	08
NERPDEST	V	X (25)	DESTINATION OF DATASET	09
NERPORIG	V	X (25)	ORIGINATOR OF DATASET	10
NERPCREA	V	X (25)	CREATOR OF NERP	34
NERPREAS	F	9 (2)	REASON CODE	35

RTR		READY TO RECEIVE	
RTRCMD	F	X(1)	COMMAND 'P'

3.4 Parameters

The following section gives details of the attributes which may be referenced by the parameters of the commands.

- All attributes are in character format.
- Format "string" refers to alpha-numeric characters:
 - . The numerals: 0 to 9
 - . The upper case letters: A to Z
 - . The following special set: / - . & () space.

However, space is not allowed as an embedded character.

- Numeric fields are always right justified and left padding with zeros must be done when needed.

3.4.01 Identification**Format: string/25 pos**





Maximum:

It identifies in a unique way the Initiator (sender) and the Responder (receiver) participating in the ODETTE FTP. The locations are adjacent for duration of the transmission session.

STRUCTURE:

Based on the ISO - IS6523: Data Interchange - Structure for the Identification of Organisations (SIO).

The SIO will consist of:

<ul style="list-style-type: none"> . ODETTE Identifier (value "O") 		1 pos
<ul style="list-style-type: none"> . Organisation Identifier (must be unique within ODETTE) 		
<ul style="list-style-type: none"> - ICD (International Code Designator) (*) (numeric) 		4 pos
<ul style="list-style-type: none"> - Organisation Code (*) (letters A to Z, digits 0 to 9, space, hyphen) 		14 pos
<ul style="list-style-type: none"> . CSA (Computer Sub-Address) (must be unique within each organisation, attributes of value will be under responsibility of the company). 		6 pos

(*) ODETTE codification Group (Group 7) will provide codes.

3.4.02 Password**Format: string/8 pos**

Maximum:

This attribute specifies the key to permit access to the adjacent locations.

Assigned by bilateral agreement.

3.4.03 Size data exchange buffer**Format: numeric/5 pos**

Maximum: 99999 Minimum: 128

This attribute specifies the maximum size of buffer that can be accepted by the location. The minimum value: 128 octets.

The size is specified in units.

The size includes the command octet. When "Special-logic" is used (see section 5), the exchange buffer size does not include the enveloping characters of the exchange buffer: STX, BSN, BCS, CR and SO-logic.

After negotiation the smallest size will be selected.

3.4.04 Send/receive capabilities**Format: alpha/1 pos**

Maximum:

This attribute specifies whether the location can SEND (S) files, RECEIVE (R) files or do BOTH (B) functions during one session.

Sending or Receiving will be serialized, so no parallel sessions will take place.

An error condition will arise if adjacent locations have the same status: send or receive.

3.4.05 Compression capabilities**Format: alpha/1 pos**

Maximum:

This attribute specifies whether the locations can handle compressed data YES (Y) or NO (N).

If one location cannot handle compressed data, option (N) will be selected after negotiation.

See section 4 for compression mechanism.

3.4.06 File name**Format: string/26 pos**

Maximum:

This is used to identify the virtual file involved in the file transfer.

The file name is assigned by bilateral agreement.

No general structure will be defined for this attribute.

FILENAME, DATE (3.4.7) and TIME (3.4.8) attributes are used to uniquely identify a virtual file.

3.4.07 Date**Format: numeric/8 pos**

Maximum:

This is the date when a file is made available for transmission at the sender's location. The DATE and TIME stamps are assigned by the file originator and have only local significance. They should not be changed by any clearing centre.

The first 4 digits (starting from the left) represent the century and the year. The 2 digits in the middle represent the month. The last 2 digits represent the day.

FILENAME (3.4.6), DATE and TIME (3.4.8) attributes are used to uniquely identify a virtual file.

3.4.08 Time**Format: numeric/10 pos**

Maximum:

This is the time when a file is made available for transmission at the sender's location. The DATE and TIME stamps are assigned by the file originator and have only local significance. They should not be changed by any clearing centre.

REFERENCE: ISO 3307.

The first 2 digits (starting from the left) define the hours.

The 2nd 2 digits represent the minutes.

The 3rd 2 digits define the seconds.

The last 4 digits is a counter (0001-9999), which gives higher resolution.

FILENAME (3.4.6), DATE (3.4.7) and TIME attributes are used to uniquely identify a virtual file.

3.4.09 Destination**Format: string/25 pos**

Maximum:

Identifies the Final Recipient of the transmission. This is the location that will look into the virtual file content and perform mapping functions. It is also the location that creates the EERP for the received file.

STRUCTURE : See parameter 3.4.1

3.4.10 Originator**Format: string/25 pos**

Maximum:

Identifies the sender of the virtual file. It is the location that created (mapped) the data for transmission.

STRUCTURE : See parameter 3.4.1

3.4.11 Maximum record size**Format: numeric/5 pos**

Maximum: 99999

This attribute specifies the maximum logical record which may be transferred to a location.

The size includes only user data and is specified in units. If the record format (3.4.25) specifies (U), this attribute must contain zeros.

3.4.12 File size**Format: numeric/7 pos**

Maximum: 9999999

This attribute specifies the amount of space that is used at the originator to store the virtual file.

The size includes only user data and is specified in K (1024) octets.

This is only a good estimated file size indication.

3.4.13 Restart position**Format: numeric/9 pos**

Maximum: 999999999

This attribute specifies the restart position for a virtual file.

The count is specified in :

- 1° Record number when the RECORD FORMAT (3.4.25) is specified as (F, V).
- 2° K octets if the RECORD FORMAT (3.4.25) is specified as (U, T).

The count will express the transmitted user data (i.e. before compression, header not included).

After negotiation between adjacent locations, retransmission will start at the lowest value.

3.4.14 Answer count**Format: numeric/9 pos**

Maximum: 999999999

See RESTART POSITION (3.4.13).

The Listener location must enter a count lower or equal to the restart count specified by Speaker location. The count will expressed the received user data.

If no restart facilities are available, a record or octet count of zero must be specified.

3.4.15 Answer reason**Format: numeric/2 pos**

Maximum:

This attribute will specify the arguments why transmission cannot proceed.

Arguments:

- 01 Invalid filename.
- 02 Invalid destination.
- 03 Invalid origin.

- 04 Storage record format not supported.
- 05 Maximum record length not supported.
- 06 File size is too big.
- 10 Invalid record count.
- 11 Invalid byte count.
- 12 Access method failure.
- 13 Duplicate file.
- 14 File direction refused
- 99 Unspecified reason.

3.4.16 Answer retry**Format: alpha/1 pos**

Maximum:

This attribute will indicate if retry is possible at a later point in time (e.g. when more storage space is available at the Listener site).

Arguments: N NO. The file should not be sent.
 Y Retry later.

This attribute is used together with ANSWER REASON code (3.4.15).

3.4.17 Record count**Format: numeric/9 pos**

Maximum: 999999999

This attribute will indicate the exact record count when RECORD FORMAT (3.4.25) was specified as (F, V). When format (U, T) was specified, this field should be zeros.

The count will express the real size of the file (i.e. before compression, header not included).

3.4.18 Unit count**Format: numeric/12 pos**

Maximum: 999999999999

This attribute will indicate the exact number of units (octets) transmitted.

The count will express the real size of the file.

3.4.19 Answer change direction**Format: alpha/1 pos**

Maximum:

This attribute will specify that change direction is requested by the Listener location.

Arguments: N No change direction.
 Y Change direction request.

3.4.20 Restart option**Format: alpha/1 pos**

Maximum:

This attribute will indicate if the location can handle partial restart of a file.

Arguments: Y Indicates YES.
 N Indicates NO.

3.4.21 SSID – Reserve**Format: string/5 pos**

Maximum:

This attribute area (in SSID command) is reserved for future use.

3.4.22 Ready message**Format: alpha/17 pos**

Maximum:

Message: "ODETTE FTP READY "

3.4.23 Carriage return**Format: char/1 pos**

Maximum:

Character with hex value "0D" or "8D".

3.4.24 Special logic**Format: alpha/1 pos**

Maximum:

This attribute specifies whether special logic (see hereafter) must be applied Y (yes) or N (no) on the data exchange buffer before the FTP moves the data into the NSDU and passes control to the network service.

3.4.25 Record format**Format: alpha/1 pos**

Maximum:

This attribute specifies the format of the virtual file.

Options are :

- F Fixed binary file.
- V Variable binary file.
- U Unstructured binary file.
- T Text file.

The attribute will be used to calculate the restart position.

3.4.26 Credit value**Format: numeric/3 pos**

Maximum: 999

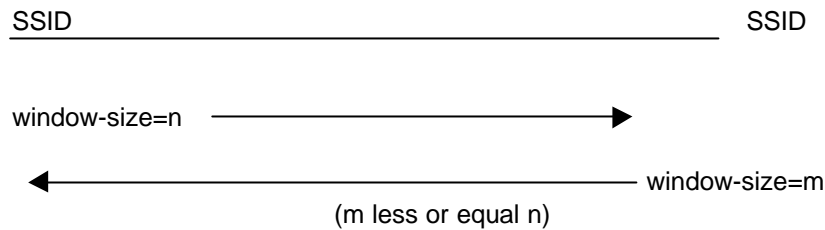
This attribute specifies the number of Data Exchange Buffers that can be sent consecutively by the Speaker. The Speaker must wait for permission of Listener (CDT) before sending more data.

In all cases, the EFID will set the value to ZERO (no data transfer allowed), since they are meaningless outside the data transfer phase. The value will be reinitialised to the negotiated value with SFID.

This value applies only to DATA flow in the Data Transfer phase.

After negotiation, the smallest size must be selected in the answer of the Responder, else a protocol error will abort the session.

Negotiation of the "credit-window-size" parameter in the SSID command:



Note: negotiated value will be "m".

3.4.27 ESID Error codes

Format: numeric/2 pos

Maximum:

- 00 Normal session termination.
- 01 Command not recognised: An NSDU has been received, where the command code (1st byte of the NSDU) could not be recognised.
- 02 Protocol violation: An NSDU was received, where the command specified a function which is invalid for the current state of the receiver.
- 03 User code not known: An SFID has been received which specified an invalid or unknown user code.
- 04 Invalid password: An SSID has been received which has specified an invalid password.
- 05 Local site emergency closedown: The local site has entered an emergency closedown mode. Communications are being forcibly terminated.
- 06 Command contained invalid data: An NSDU has been received, which has specified a valid command and which had a valid length. However, a field within the NSDU contains invalid data.
- 07 NSDU size error: An NSDU has been received which has a valid command, but it's length is incorrect for the command specified.
- 08 Resources not available: The request for connection has been denied because no resources are available. The connection attempt should be retried later.
- 09 Time out.
- 10 Mode or capabilities incompatible.
- 99 Unspecified abort code: An error was detected that cannot be adequately described by the codes below.

3.4.28 Protocol release level

Format: numeric/1 pos

Maximum:

Used to specify the level of the protocol. It should be set to the constant '4' for Revision 1.4

'2' for Revision 1.3

'1' for Revision 1.2

. The protocol release level is negotiable in the same way as CREDIT VALUE (3.4.26) attribute, etc ... and the lowest version of the protocol is selected.

3.4.29 SSID – User field**Format: string/8 pos**

Maximum:

This field (In SSID command) can be used by the FTP in any way. If it is not used then it should be initialized to spaces.

It is intended for use between two FTP's where a special bilateral agreement and understanding exists as to the meaning of the user field.

3.4.30 SFID/EERP – Reserve**Format: string/3 pos**

Maximum:

This attribute area (in SFID or EERP commands) is reserved for future use.

3.4.31 SFID/EERP – User field**Format: string/8 pos**

Maximum:

This field (In SFID or EERP commands) can be used by the FTP in any way. If it is not used then it should be initialized to spaces.

It is intended for use between two FTP's where a special bi-lateral agreement and understanding exists as to the meaning of the user field.

3.4.32 CDT – Reserve**Format: string/2 pos**

Maximum:

This attribute area (in CDT command) is reserved for future use.

3.4.33 NERP – Reserve**Format: string/6 pos**

Maximum:

This attribute area (in NERP command) is reserved for future use.

3.4.34 NERP – Creator of NERP**Format: string/25 pos**

Maximum:

Identifies the creator of the NERP. It is the location that could not send the virtual file to the next recipient.

STRUCTURE : See parameter 3.4.1

3.4.35 NERP – Reason code**Format: numeric/2 pos**

Maximum:

This attribute will specify the arguments why transmission cannot proceed. Because the cause is due to a received ESID or SFNA, it mainly maps these codes to the newly created NERP reason code.

Arguments:

- 03 Reception of ESID with reason code '03' (user code not known).
- 04 Reception of ESID with reason code '04' (invalid password).
- 09 Reception of ESID with reason code '99' (unspecified abort code).
- 11 Reception of SFNA (RETRY=N) with reason code '01' (invalid filename).
- 12 Reception of SFNA (RETRY=N) with reason code '02' (invalid destination).
- 13 Reception of SFNA (RETRY=N) with reason code '03' (invalid origin).
- 14 Reception of SFNA (RETRY=N) with reason code '04' (invalid storage record format not supported).
- 15 Reception of SFNA (RETRY=N) with reason code '05' (maximum record length not supported).
- 16 Reception of SFNA (RETRY=N) with reason code '06' (file size too big).
- 20 Reception of SFNA (RETRY=N) with reason code '10' (invalid record count).
- 21 Reception of SFNA (RETRY=N) with reason code '11' (invalid byte count).
- 22 Reception of SFNA (RETRY=N) with reason code '12' (access method failure).
- 23 Reception of SFNA (RETRY=N) with reason code '13' (duplicate file).
- 24 Reception of SFNA (RETRY=N) with reason code '14' (file direction refused)
- 50 Transmission stopped by operator.
- 99 Unspecified abort code: An error was detected that cannot be adequately described by the codes below.

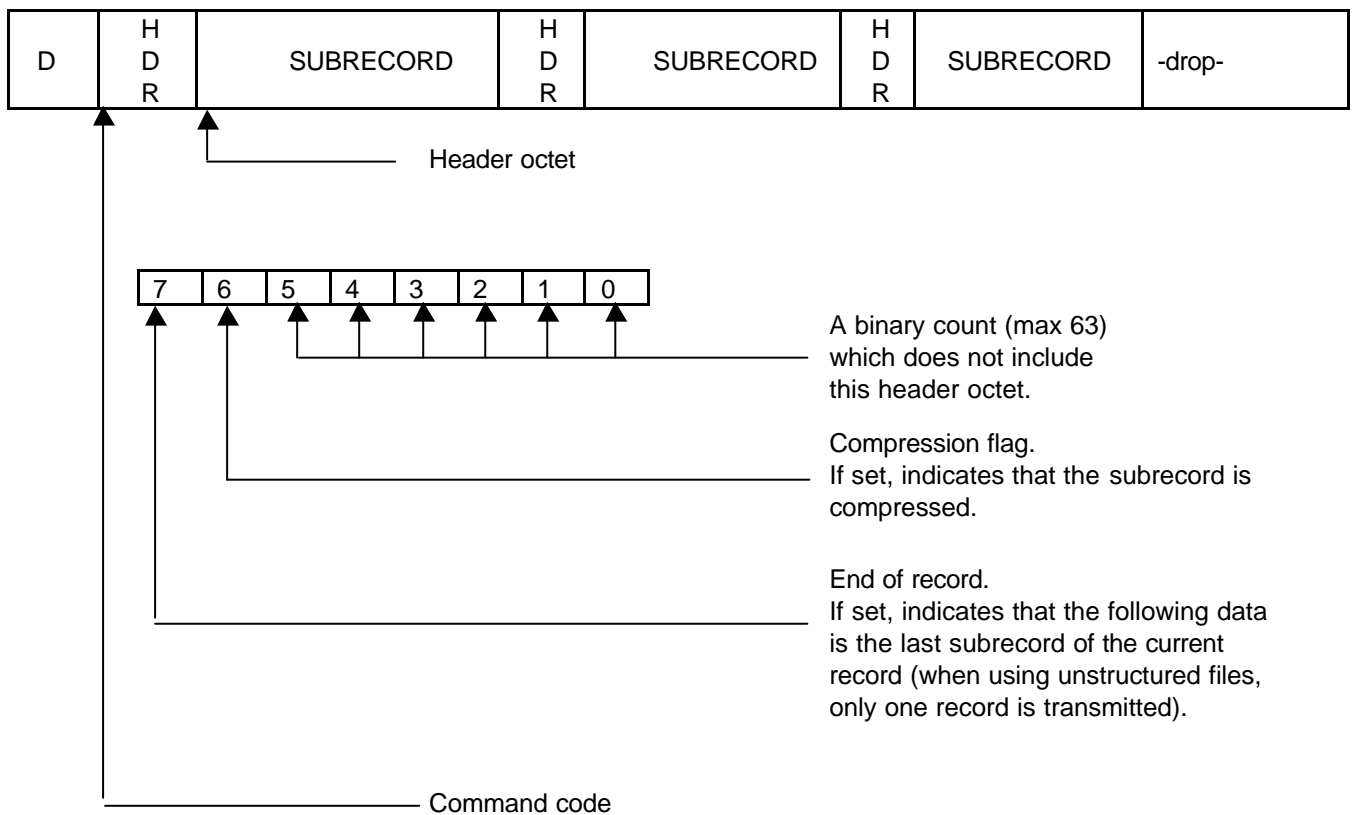
IV. Data exchange buffer

(Based on : NIFTP by the High Level Protocol Group, published: February 5, 1981)

The FTP will map the virtual file record length into the DATA EXCHANGE BUFFER specified at session start phase.

All data (virtual file) flow back and forwards via the Data Exchange Buffer.

DATA EXCHANGE BUFFER STRUCTURE:



The data exchange buffer structure permits records of arbitrary length and permits data compression. So, a record length may be:

- a) Lower than the exchange buffer's length.
- b) Equal to the exchange buffer's length.
- c) Greater than the exchange buffer's length.

In case (a), concatenation of records within one exchange buffer may take place.

In case (c), the record will be mapped into multiple exchange buffers.

If the buffer is not completely full, truncated buffers will be transmitted.

For transmission of virtual file records, data is divided into subrecords, each of which is preceded by a header octet. A subrecord is never expanded over two exchange buffers. So if the remaining part of an exchange buffer is not sufficient for a "complete" subrecord, then the two following ways are allowed:

- 1) Either truncate the first exchange buffer, and put the complete subrecord (preceded by its header octet) in a second exchange buffer.
- 2) Or use one subrecord just filling the first exchange buffer, and create the other needed subrecord(s), preceded by the header octet(s), in the second exchange buffer.

A record can be of length equal to zero, it may appear anywhere in the exchange buffer. A subrecord can be of length equal to zero, it may appear anywhere in the record and/or in the exchange buffer.

If no special logic is required the FTP moves the Data Exchange Buffer into the NSDU and passes control to the network service.

If the compression flag is set, then the count gives the number of times that the single octet following the header must be inserted when constructing the compressed virtual file record.

EXAMPLE:

As an example of an exchange buffer being filled as per the above, the following text will serve as an illustration. Each line of the text will be considered a "record". A blank line is only used for presentation purposes and it is not part of the exchanged data.

It is an ancient Mariner,
And he stoppeth one of three.
"By thy long grey beard and glittering eye,
"Now wherefore stopp'st thou me ?

"The bridegroom's doors are opened wide,
"And I am next of kin;
"The guests are met, the feast is set :
"May'st hear the merry din."

He holds him with his skinny hand,
"There was a ship," quoth he.
"Hold off ! unhand me, grey-beard loon !"
Eftsoons his hand dropt he.

He holds them with his glittering eye -
The Wedding-Guest stood still,
And listens like a three years' child :
The Mariner hath his will.

The Wedding-Guest sat on a stone :
He cannot chuse but hear ;
And thus spake on that ancient man,
The bright-eyes Mariner.

The ship was cheered, the harbour cleared,
Merrily did we drop
Below the kirk, below the hill,
Below the light-house top.

The exchange buffers below were built from the above. The top line of each represents the ASCII code, whilst the two lines below give the hexadecimal value.

Remember that :

- . The "D" at the beginning of each exchange buffer is the command code.
- . The "." preceding each subrecord is the header octet (see the hexadecimal value).

Exchange buffer 1

D₂It is an ancient Mariner,₂And he stoppeth one of three.₂"By th
494726726626666667246766672946626627767767626662662767662A247276
499409301E01E395E40D129E52CD1E4085034F005480FE50F6048255EB229048

y long grey beard and glittering eye.₂"Now wherefore stopp'st th
726662676726667626662666776766626762A2467276676667627767727276
90CFE70725902512401E407C944529E70595C12EF70785256F25034F00734048

ou me ?₂"The bridegroom's doors are opened wide.₂"And I am next
6726623A25662676666766627266677267626766666276662924662426626677
F50D50F9248502294572FFD7304FF2301250F05E45407945C621E40901D0E584

of kin.₂"The guests are met, the feast is set :₂"May'st hear th
26626663A25662676772767626667227662666772672767230246727726667276
0F60B9EB72485075534301250D54C048506513409303540AF2D1973408512048

Exchange buffer 2

D₂e merry din."₂He holds him with his skinny hand.₂"There was a
486266777266622A462666672666276762667276666726666292566762767262
4D50D5229049EE228508FC43089D07948089303B9EE9081E4CD2485250713010

ship," quoth he.₂"Hold off ! unhand me, grey-beard loon !" .Eftso
7667222776762662A24666266622276666626622676726667626666222946776
3890C2015F48085E928FC40F660105E81E40D5C07259D251240CFFE012B5643F

ons his hand dropt he.₂He holds them with his glittering eye -₂T
6672667266662676772662A46266667276662767626672666776766626762295
FE30893081E4042F04085E78508FC430485D07948089307C944529E705950DE4

he Wedding-Guest stood still,₂And listens like a three years' ch
6625666666247677277666277666224662667766726666262767662766772266
85075449E7D75534034FF40349CCC21E40C9345E30C9B5010482550951237038

Exchange buffer 3

D₂ild :₂The Mariner hath his will.₂The Wedding-Guest sat on a st
4866623956624676667266762667276662A56625666666247677276726626277
459C40AA4850D129E52081480893079CCE2485075449E7D7553403140FE01034

one :₂He cannot chuse but hear ;₂And thus spake on that ancient
66623946266666726677626772666723A4662767727766626627667266666672
FE50AA85031EEF40385350254085120B31E4048530301B50FE0481401E395E40

man.₂The bright-eyes Mariner.₂The ship was cheered, the harbour
66629566267666726767246766672A5662766727672666676622766266766772
D1EC84850229784D59530D129E52EA48503890071303855254C048508122F520
cleared,₂Merrily did we drop.₂Below the kirk, below the hill,₂Bel
66667662946776672666276267679466672766266726626666727662666620466
3C51254C3D5229C90494075042F0F25CF704850B92BC025CF70485089CCC325C

Exchange buffer 4

D₂ow the light-house top.
4967276626666726677627672
47F704850C9784D8F53504F0E

V. Special logic

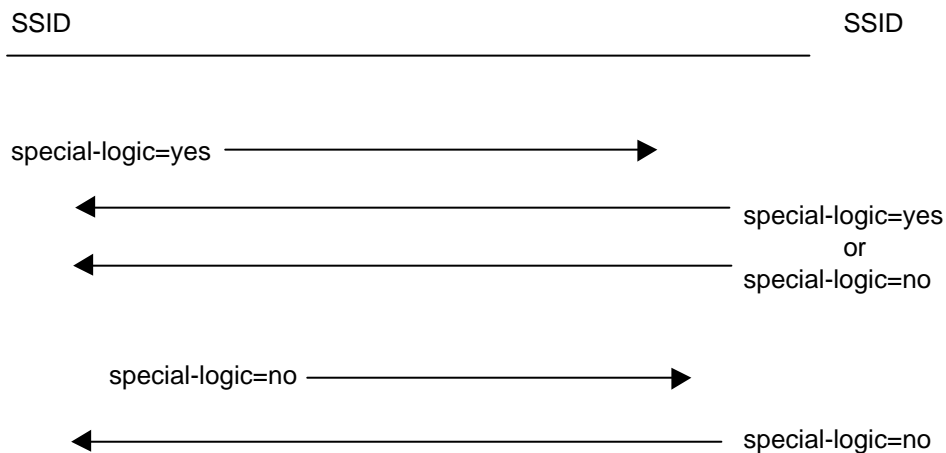
5.1 When special logic is not to be used

This logic is not applied on SSRM and SSID commands.

5.2 The need for "enveloping" exchange buffers

The "special-logic" was created in order to allow the use of FTP over asynchronous links. The "special-logic" could be needed to enable terminals to access a X25 network via an asynchronous entry (through a PAD: Packet Assembly / Disassembly). The "special-logic" is not needed in case of a whole X25 connection. This "special-logic" realises a CRC function in order to detect errors due to the asynchronous medium.

Negotiation of the "special-logic" parameter in the SSID command:



This logic is activated when the SPECIAL LOGIC parameter in the SSID specifies Y (yes).

Special logic processing, when activated, will function within level 4 of the OSI model.

Level-7 FTP	application	←	FILE SERVICE
Level-6 FTP	presentation		
Level-5 FTP	session		
Level-4 FTP	transport	←	NETWORK SERVICE
	SPECIAL LOGIC PROCESSING		
Level-3	X.25		
Level-2	X.25		
Level-1	X.25		

5.3 Responsibilities of special logic

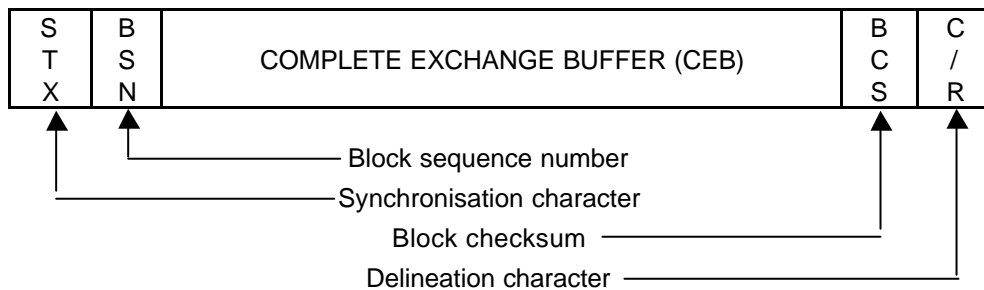
When transmitting an exchange buffer and special logic is active, layer 4 will wrap the exchange buffer in synchronization and delineation characters, then protect the data integrity by means of a block checksum (BCS).

When receiving an exchange buffer and special logic is active, layer 4 will remove such things as synchronization and delineation characters etc ... before passing the exchange buffer to the higher layers.

5.4 Extended exchange buffer format

Each envelope has one byte header prefixed to it, and a 2 bytes checksum appended to the end. The checksum is derived in a manner specified in the ISO DIS 8073 TRANSPORT LAYER documentation.

The layout of the data buffer will be structured as follows:



The envelope is initialized with a STX and the checksum variables are set to 0. The leading STX is not protected by the checksum calculation but is explicitly protected by a character compare at the receiver's end. The exchange buffer is processed character by character. As each character is removed from the exchange buffer it is put through the checksum calculation and then, prior to its insertion in the envelope it is put through the Shift-out transparency logic, which will result in either one or two characters being inserted. When the contents of the exchange buffer have been entirely processed then the checksum variables are brought up to date by inserting two X'00's through the checksum calculator and the two resultant checksum characters forwarded to the shift-out transparency logic for insertion into the envelope. Finally a carriage return (CR) is appended to the envelope. The segment is now ready for transmission to line.

Upon receipt of a valid envelope that has the correct sequence number, the host should increment his sequence number register ready for the next transmission.

The receiver will initialize his receiving buffer area upon receipt of a STX character, place the STX at the beginning of the buffer and reset checksum variables. All subsequent characters are processed using Shift-out logic before they are inserted into the buffer, at which point they will be NOT processed by the checksum calculator, although the character following the Shift-out (after subtracting X'20') will be. The

checksum characters themselves will be processed by the checksum calculator by virtue of the design of the checksum algorithm.

Synchronization is the state an OFTP entity is in after receipt of an STX.

Synchronization mode continues unless terminated by receipt of a C/R or expiration of any current active timer.

5.5 Error recovery

5.5.1 Mechanism

The error correction scheme is implemented by the definition of three Timers and the use of a ASCII NAK (Negative Acknowledgement) character followed by a C/R. The <NAK><C/R> will flow between the two session partners, but only as a consequence of previous bad data.

User of the error recovery correcting extension must always work with a Credit Value of 1. This can be forced upon any session partner at SSID negotiation. The effect will be to force simple half-duplex flip-flop protocol.

Upon receipt of a bad block, send <NAK><C/R> to the session partner.

Upon receipt of a <NAK><C/R>, a session partner should retransmit the last block in its entirety.

5.5.2 Timers

The majority of error conditions will be detected by a bad BCS sequence. However, certain conditions cannot be so detected. For example, a corrupt C/R will mean that the receiver will not know that the end of a block has been reached. No matter how long he waits, no more data will come from the sender. Thus a Timer is the only way to detect this type of corruption. There are three Timers needed to detect all possible malignant conditions of this type.

- T1 - Exchange Buffer Time Out (Inactivity or Response)
- T2 - Inter Character Time Out
- T3 - Data Carrier Detect Loss Time Out

The three Timers are in addition to the timer defined in the original protocol.

TIMER T1 - RESPONSE TIME OUT (DEFAULT = 45 SECONDS):

Used to detect a high level block Time Out. E.g. the Time Out between an SFID and its associated SFPA or SFNA response.

- | | |
|-----------|---|
| Started - | It is started after the last character of an exchange buffer has been sent to the line. |
| Stopped - | It is stopped when a STX has been received. |
| Expiry - | Retransmit the whole block again, until such time as the retry limit has been reached. |

TIMER T2 - INTER CHARACTER TIME OUT (DEFAULT = 7 SECONDS):

Used to detect errors in the reception of individual characters.

Started - For an asynchronous entity it is started upon receipt of each character whilst in synchronisation mode.

For an X.25 entity it is started after a received block that did not terminate an exchange buffer.

Stopped - Upon receipt of the next character.

Expiry - Send a <NAK><C/R>, drop out of synchronised mode and go back and listen to line.

TIMER T3 - DATA CARRIER TEMPORARY LOSS (DEFAULT = 1 SECOND):

Used by an asynchronous entity only and is used to detect a temporary carrier failure.

Started - When DCD (Data Carrier Detect) is lost.

Stopped - When DCD is regained.

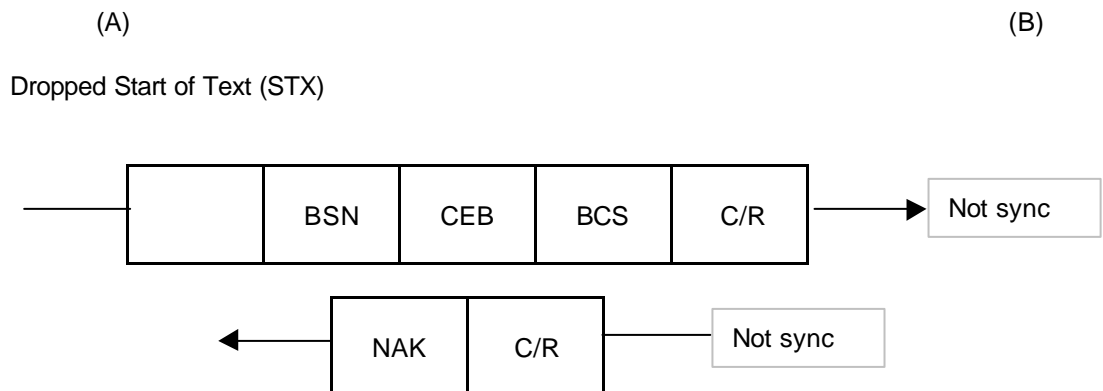
Expiry - Disconnect the session.

5.5.3 Types of error

Data corruption when it occurs can be categorised in any one of five ways:

(1) CORRUPT STX (START OF TEXT)

In this situation the STX is not seen and synchronisation is not achieved. The terminating C/R is received out of synchronisation and hence the block is not seen by the receiver. A <NAK><C/R> is transmitted to the sender to indicate this. The sender should then retransmit the last block (each implementation will need to set a retry limit to be used for the number of consecutive times it attempts to retransmit a block - a default limit of 5 is recommended). All data received outside synchronisation (except <NAK><C/R>) are ignored.



Exchange Buffer Resent

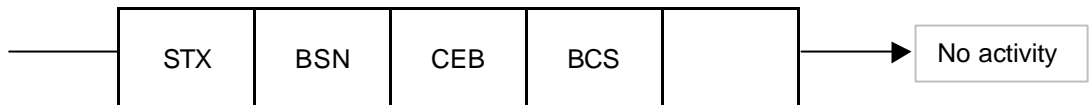


(2) CORRUPT TERMINATION (C/R)

This situation manifests itself as an extended period of synchronisation with no activity. The T2 Timer will detect this condition.

(A)

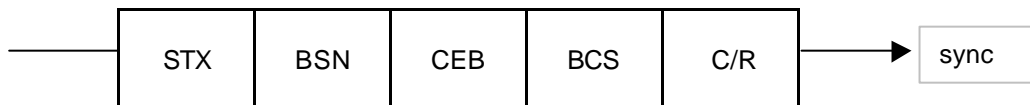
(B)



Corrupt Carriage Return



Exchange Buffer Resent



(3) BAD DATA

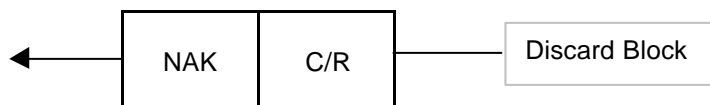
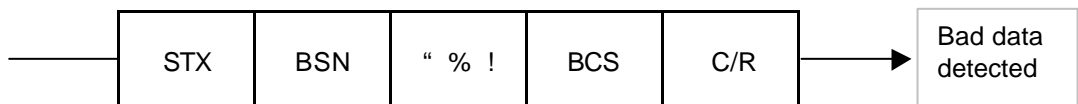
(4) BAD BCS (BLOCK CHECK SUM)

In this situation, the receiver is unable to know whether the error is bad data, or bad BCS. In either case the response is to discard the exchange buffer and send a <NAK><C/R>.

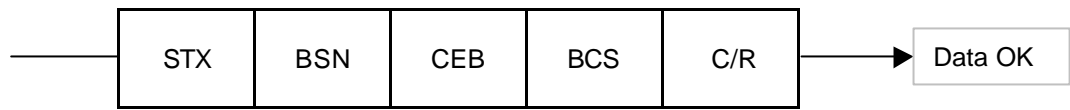
(A)

(B)

Bad Data/BCS



Exchange Buffer Resent



(5) BAD BLOCK SEQUENCE NUMBER (BSN)

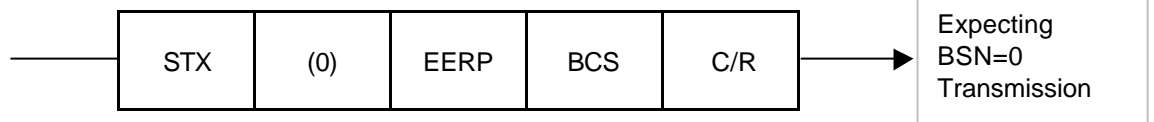
A circular sequential number (0 up to and including 9) is assigned to transmitted exchange buffers. This is to aid detection of duplicate or out of sequence exchange buffers. Once a duplicate block is detected, the exchange buffer in question is discarded. Once an out of sequence block is detected this should result as a protocol violation.

Example protocol sequence:

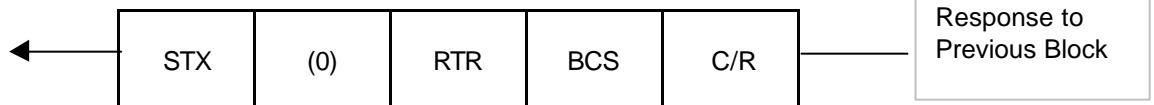
(A)

(B)

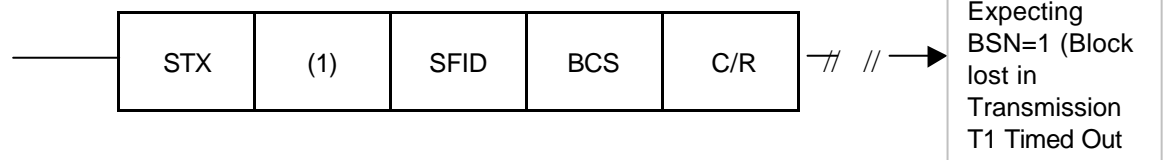
Exchange Buffer Being Sent



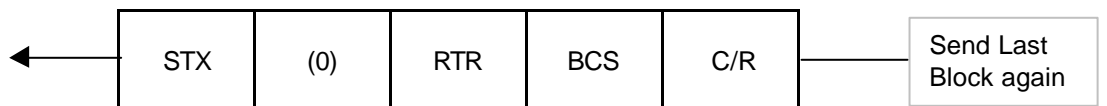
Exchange Buffer Being Sent



Exchange Buffer Being Sent



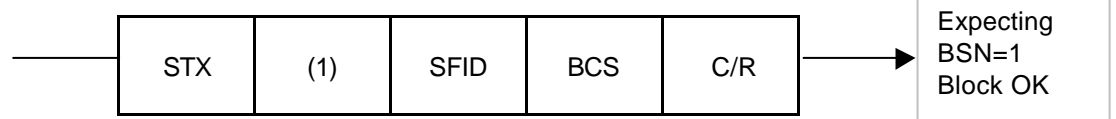
Exchange Buffer Being Sent

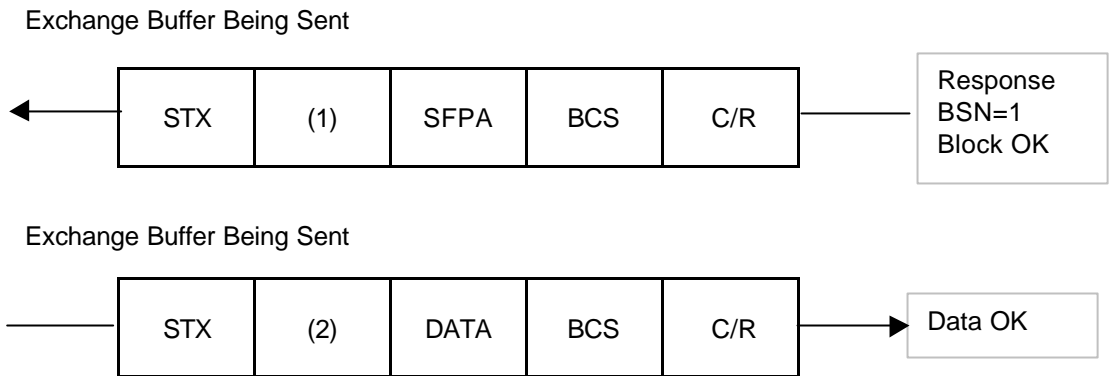


Discard Block and start Timer T1

T1 Timed Out

Exchange Buffer Resent





Note: A credit value of 1 must be used to guarantee half-duplex flip-flop.

5.6 Sequence of events for special logic processing

Following functions will be executed in sequence:

- 1 Calculation of the Block Sequence Number (BSN):

BSN is set to zero by SSID. First block will be sent with value zero. Value of BSN is increased by one for each data buffer to be transmitted. When BSN value exceeds 9, counter will be reset to zero.

Format: numeric/1 pos.

- 2 Calculation of the Block CheckSum (BCS):

Calculation is done as specified in the ISO DIS 8073 TRANSPORT LAYER document.

Format: binary/2 pos.

- 3 Shift-out transparency (See TRANSMIT/RECEIVE logic)

To avoid appearance of any control characters in the data stream, all the characters of the extended exchange buffer (with exception of the STX and carriage return characters enveloping the buffer) are put through a Shift-out logic, which result in a character being inserted (SO) and adding hex value '20' to the control character.

- 4 The carriage return is inserted at the end of the data buffer.

NOTE: After adding STX, BSN, BCS, CR and SO-logic, the data buffer may exceed the data exchange buffer size.

5.7 Checksum creation algorithm

These follow the ISO DIS 8073 TRANSPORT LAYER standard.

SYMBOLS:

The following symbols are used:

C0,C1 Variables used in the algorithm
L Length of the complete TPDU
X Value of the first octet of the checksum parameter
Y Value of the second octet of the checksum parameter

ARITHMETIC CONVENTIONS:

Addition is performed in one of the two following modes:

- a) modulo 255 arithmetic,
- b) one's complement arithmetic in which if any of the variables has the value minus zero (i.e. 255) it shall be regarded as though it was plus zero (i.e. 0).

ALGORITHM FOR GENERATING CHECKSUM PARAMETERS:

- . Setup the complete TPDU with the value of the checksum parameter field set to zero.
- . Initialize C0 and C1 to zero.
- . Process each octet sequentially from i=1 to L by
 - a) adding the value of the octet to C0; then
 - b) adding the value of C0 to C1.
- . Calculate X and Y such that
$$X = C0 - C1$$
$$Y = C1 - 2 * C0$$
- . Place the values X and Y in the checksum bytes 1 and 2 respectively.

5.8 Algorithm for checking checksum parameters

- . Initialize parameters C0 and C1 to zero.
- . Process each octet of TPDU sequentially from i=1 to L by
 - a) adding the value of the octet to C0; then
 - b) adding the value of C0 to C1.
- . If, when all the octets have been processed, either or both C0 and C1 does not have the value zero, then the checksum formulas have not been satisfied.

Note that the nature of the algorithm is such that it is not necessary to compare explicitly the stored checksum bytes.

5.9 Shift-out processing

(Transparency for all control characters)

TRANSMIT LOGIC (values SO: X'0E' or X'8E')

Buffer(1), ... , (n) is a character in the buffer to be sent.

```
FOR i=1 to n          /* for all octets of the buffer */
IF ((buffer(i) & X'7F') < X'20')
```

```
THEN output (SO)
output (buffer(i) + X'20')
```

```
ELSE output (buffer(i))
```

NEXT:

RECEIVE LOGIC (values SO: X'0E' or X'8E')

Buffer(1), ... , (n) is a character in the received buffer.

```
drop = false
FOR i= 1 to n          /* for all octets of the buffer */
IF drop = true
```

```
THEN output (buffer(i) - X'20')
drop = false
```

```
ELSE IF buffer(i) = (X'0D' or X'8D')
```

```
THEN Stop
```

```
ELSE IF buffer(i) = SO
```

```
THEN drop = true
```

```
ELSE output (buffer(i))
```

NEXT:

5.10 PAD Parameter profile

Before a (OFTP) asynchronous entity -->Modem-->PAD--> (OFTP) native X.25 link can be established, the target PAD parameters must be set such that correct communication is established. It is strongly recommended that the PAD-parameters are set by the X.25 entity. CCITT recommendations X.3, X.28 and X.29 define the PAD parameters and procedures for exchange of control information and user data between a PAD and a packet mode DTE.

Following is the Parameter list and values used to set the PAD for OFTP communication. For further detailed information see the specification for CCITT X.25, X.28, X.29 and X.3.

No	Description	Value	Meaning
1	Escape from Data Transfer	0	Controlled by host
2	Echo	0	No Echo
3	Data Forwarding Signal	2	Carriage Return
4	Selection of Idle Timer Delay	20	1 second
5	Ancillary Device Control	0	X-ON, X-OFF not used
6	PAD Service Signals	1	All except prompt
7	Procedure on Break	2	Reset
8	Discard Output	0	Do not discard
9	Padding after Carriage Return	0	No padding
10	Line Folding	0	No line folding

11 Terminal Data Rate	-	Read only
12 Flow Control of the PAD	0	No flow control used
13 Linefeed Insertion after C/R	0	No line feed
14 Linefeed Padding	0	No line feed padding
15 Editing	0	No editing
16 Character Delete	127	Delete
17 Line Delete	24	<CTRL>X
18 Line Display	18	<CTRL>R
19 Editing PAD Service Signals	0	No service signal
20 Echo Mask	0	No echo mask
21 Parity Treatment	0	No parity check
22 Page Wait	0	No page wait

Note 1:

Refer to CCITT (1984)

- Parameter 1 - 12 are mandatory and available internationally.
- Parameter 13 - 22 may be available on certain network and may also be available internationally.
- A parameter value may be mandatory or optional.

The Odette profile refers only to parameter values which must be internationally implemented if the parameter is made available internationally.

The OFTP special logic option may be impossible on some PAD:s because of none support of some of the parameters (13 - 22). (If the PAD is supporting parity check (21) by default, OFTP special logic would be impossible.)

It is a user responsibility to ensure special logic consistency when making the PAD subscription.

Note 2:

Some parameters may have to be set differently depending on:

- Make and function of the start-stop mode DTE entity.
- Start-stop mode DTE entity OFTP monitor function.
- PAD services implemented.
- Packet mode DTE entity OFTP monitor function.

VI. State transition tables and diagrams

6.1 Input events

DESTIN.	ABREV.	NAME	PARAMETERS
Internal	TIME-OUT	inactivity timer runs out	none
Network	N_CON_IND	Network Connect. Ind.	called-address, calling address, user-data
	N_CON_CF	Network Connect. Conf.	none
	N_RST_IND	Network Reset Ind.	none
	N_DATA_RQ	Network Data Ind.	data contains 1 F_PDU
	N_DISC_RQ	Network Disconnect. Ind.	none
F_PDUs	SSID	Start Session	see section 3
	SFID	Start File	“ “ “
	SFPA	Start File Positive answ.	“ “ “
	SFNA	Start File Negative answ.	“ “ “
	EFID	End of File	“ “ “
	EFPA	End of File Positive answ.	“ “ “
	EFNA	End of File Negative answ.	“ “ “
	DATA	Data	“ “ “
	ESID	End of Session	“ “ “
	EERP	End to End Response	“ “ “
	NERP	Negative End Response	“ “ “
	RTR	Ready to Receive	“ “ “
	CD	Change Direction	“ “ “
	CDT	Set credit	“ “ “
	SSRM	Ready Message	“ “ “
	User	F_CONNECT_RQ	Connection request
F_CONNECT_RS		Connection response	“ “ “
F_ABORT_RQ		Abort Request	none
F_RELEASE_RQ		End Session Request	none
F_START_FILE_RQ			see service definition
F_START_FILE_RS(+)			“ “ “
F_START_FILE_RS(-)			“ “ “
F_DATA_RQ			“ “ “
F_CLOSE_FILE_RQ			“ “ “
F_CLOSE_FILE_RS(+)			“ “ “
F_CLOSE_FILE_RS(-)			“ “ “
F_CD_RQ			none
F_EERP_RQ		End-to-End Response	file-name, date, time, destination, origin.
F_NERP_RQ			see service definition

Note: N_RST_CF is never used since N_RST_RQ is never sent to the Network-Service.

6.2 Output events

DESTIN.	ABREV.	NAME	PARAMETERS
Network	N_CON_RQ	Network Connect. Req.	called-address, calling address, user data
	N_CON_RS		none
	N_DATA_RQ	Network Connect. Resp.	one F_PDU
	N_DISC_RQ	Network Data Req. Network Disconnect. Req.	none
Peer FTP Entity		See input events and their parameters Entity	
User	F_CONNECT_IND		see service definition
	F_CONNECT_CF		“ “ “
	F_ABORT_IND		“ “ “
	F_RELEASE_IND		“ “ “
	F_START_FILE_IND		“ “ “
	F_START_FILE_CF(+)		“ “ “
	F_START_FILE_CF(-)		“ “ “
	F_DATA_IND		“ “ “
	F_CLOSE_FILE_IND		“ “ “
	F_CLOSE_FILE_CF(+)		“ “ “
	F_CLOSE_FILE_CF(-)		“ “ “
	F_EERP_IND		“ “ “
	F_NERP_IND		“ “ “
F_CD_IND		“ “ “	

Note: N_RST_RS not used

6.3 States

ABREV	NAME	COMMENTS
IDLE I_WF_NC I_WF_RM I_WF_SSID	Connection IDLE Init waiting for Network connection Init waiting for Ready Message Init waiting for SSID	 before sending the SSID, the initiator will wait for a Ready Message sent by the responder. After having sent SSID
A_NC_ONLY A_WF_CONRS	Respond, N_C just opened Respond, waiting for F_CONNECT_RS	Wait for SSID After having received SSID
WF_NDISC	Waiting for N_DISC_IND	After having sent ESID
WFCD ERSTWFCD NRSTWFCD	Waiting for CD EERP stored while in WFCD state NERP stored while in WFCD state	This state is entered when the local user must become the speaker, and a CD is awaited at the protocol level EFPA (rquest-cd=yest) sent by listener. Since the WFCD state is not viewed by the user, the user may send F_EERP_RQ, F_START_FILE_RQ or F_CD_RQ before the FTP receives CD command Since the WFCD state is not viewed by the user, the user may send F_NERP_RQ before the FTP received CD command.
WFRTR	Waiting for RTR	EERP or NERP sent, waiting for RTR
SFSTWFCD CDSTWFCD	Start File RQ stored while in WFCD state	Since the WFCD state is not viewed by the user, the user may send F_START_FILE_RQ before the FTP received CD command.
IDLESP IDLESPCD OPOP OPO OPOWFC CLOP	Idle Speaker Idle Speaker and CD_IND has been passed to the user Open out pending Open out (data regime) Open out waiting for credit Close out pending	The user may either start a new file or issue F_CD_RQ F_RELEASE_RQ possible but not F_CD_RQ SFID SENT, SFPA/NA awaited SFPA received, F_DATA_RQ or F_CLOSE_FILE_RQ are awaited emission credit counter "CDTCE" comes to zero while data transmission phase. EFID sent, EFNA/PA awaited
IDLELI	Idle listener	
IDLELICD	Idle listener after CD_RQ from user	receiving ESID valid
OPIP	Open input pending	SFID received, F_START_FILE_RS awaited
OPI	Open input (data regime)	DATA FPDU awaited
CLIP	Close input pending	EFID received, wait F_CLOSE_FILE_RS

6.4 Conventions for state tables

6.4.1 State table notation

The intersection of a given INPUT EVENT and a given STATE defines the action that the FTP entity will do when receiving the event in this state.

The FTP action is formally defined as follows:

FTP-ACTION ::= NOT-POSSIBLE USER-ERROR TRANSITION-LIST.

NOT-POSSIBLE ::= "X" /* a cross over the box indicates that this case is not possible. See below for semantics */

USER-ERROR ::= "UE" /* the semantics for user error is given below */

TRANSITION-LIST ::= SINGLE-TRANSITION EXTENDED-TRANSITIONS .

SINGLE-TRANSITION ::= BLOC TO-STATE ";"

BLOC ::= ACTION OUTPUT-EVENT nothing .

ACTION ::= "(" ACTION-LIST ")" .

ACTION-LIST ::= SINGLE-ACTION SINGLE-ACTION "," ACTION-LIST

SINGLE-ACTION ::= "1" "2" "3"

/* the number refers to an action table */

TO-STATE ::= /* one state defined in section 6.3 */

OUTPUT-EVENT ::= /* one event defined in section 6.2 */

EXTENDED-TRANSITIONS ::= PREDICATE SINGLE-TRANSITION EXTENDED-TRANS.

PREDICATE ::= BOOL-EXPR ":" .

BOOL-EXPR ::= /* a boolean expression where the operands are of the form Pxx, where xx is a number referring to an entry into a predicate table */

At any time, comments may be introduced. They are noted:

(" NOTE-LIST ") .

NOTE-LIST ::= number number "," NOTE-LIST.

/* number refers to an entry into a table containing the comments */

According to the following notation a state table is made of:

- A predicate table,
- An action table,
- A notes/comments table,
- The state table itself.

6.4.2 Example

In order to facilitate the understanding of the notation, we give hereafter an example and its interpretation.

Predicate table:

entry number	predicate definition
1	Resources are available
2	Addresses are valid
etc...	

Action table:

entry number	action definition
1	Store in local variable V.id the i.d parameter of the received F_CONNECT_RQ primitive.
2	Set parameters of N_CON_RQ (called and calling addresses) with values received in F_CONNECT_RQ.
etc...	

Notes and comments table:

entry number	comment definition
1	V.id will be used when N_CONNECT_CF is received to set the id parameter of SSID
etc...	

State table:

STATE \ EVENT	IDLE	OTHER STATES
F_CONNECT RQ	(P1,P2) : ((1,2)) (1) N_CON_RQ I_WF_NC; not (P1,P2) : F_ABORT_IND IDLE	
Other events		

SEMANTICS:

```

If (resources available and addresses valid)
  then begin    V.id=F_CONNECT_RQ.id; /* will be used in SSID */
               out N_CON_RQ(F_CONNECT_RQ.called-addr,
                           F_CONNECT_RQ.calling-addr)
               next state = I_WF_NC; end
  else begin   out F_ABORT_IND
               next state = IDLE; end

```

6.4.3 Referencing variables in predicates and actions

There are 3 kind of data which need to be used in predicates and actions:

1° Local variables:

Variables locally kept by the FTP entity in order to help the protocol operation.

They are referred to by "V.variable-name".

2° Local constants:

They define capability of the FTP entity (i.e. the entity which supports a given feature).

They are referred to by "C.constant-name".

3° Parameters of events:

They correspond to the parameters of the input and output events. They are referred to as:

- "I.event-name.parameter-name" for input events.
- "O.event-name.parameter-name" for output events.

Constants and input events parameters are only checked (i.e. never assigned), while variables can be either tested or assigned.

6.4.4 Handling of errors in state table

Receiving an event into a given state may be invalid for 3 reasons:

- 1° The case is really impossible by construction of the state automata:
For instance a timer which has not been set cannot run out. This is noted "X" (a cross over the box).
- 2° Receiving this event in this state would be an error of the Network-Service provider:

This is also noted "X"; since we consider that the local implementation of the Network-Service is correct.
- 3° Receiving this event in this case is a user error noted "UE":

The state tables define what are the conditions for a user event to be valid. This prevents the generation of protocol errors as results of user errors.

But the state tables DO NOT define what is the reaction of the FTP in the case of a user error: It is a matter local to a FTP implementation to prohibit user error by any means (e.g. ignoring the event, or generating an abort or any other means).

The state tables also define the conditions under which the receipt of PDU is valid (i.e. detection of protocol errors).

The state tables also DO define the reaction of an FTP when receiving a protocol error.

6.5 Definition of local variables used by the FTP

VARIABLE-NAME	TYPE	COMMENTS	INITIAL VALUE
V.buf-size	integer	The exchange-buffer-size as agreed	none
V.compression	YES/NO	Compression in used as agreed	none
V.restart	YES/NO	Restart in used as agreed	none
V.mode		Sender-only, Receiver-only, Both	none
V.restart-pos	integer	Used only during file opening	none
V.id	string	For building O.SSID.id	none
V.called-add	address	For building O.F_CONNECT_IND.called-add	none
V.calling-add	address	For building O.F_CONNECT_IND.calling-add	none
V.req.buf	primitive	For storing an input event (F_XXX_RQ) in WFCD	none
V.window	integer	The value of credit adopted and negociated during the whole session	none
V.logic	YES/NO	YES: indicates that the special logic was agreed by both entities during the current session.	none

6.6 Definition of local constants used by the FTP

CONSTANT-NAME	COMMENTS	VALUE
C. cap-compression	Whether the FTP supports compression or not	YES or NO
C.cap-mode	The mode supported by the FTP entity - supports sender-only - supports receiver-only - supports both	sender-only receiver-only both
C.init-cap	Whether the FTP entity can be: - initiator - responder - both	I R B
C.buf-size	The maximum buffer size supported by the entity	127 < int. < 100000
C.cap-logic	Whether the FTP does not support the special logic, or it can support it, or it needs it. - does not support special-logic - supports special-logic - needs special-logic	0 1 2
C.window	local maximum/chosen credit value	integer < 1000

6.7 State tables

They are divided into 5 state tables:

- 1° TABLE 1 : Session connection phase
- 2° TABLE 2 : Error cases and abort
- 3° TABLE 3 : Speaker part-1
- 4° TABLE 4 : Speaker part-2
- 5° TABLE 5 : Listener

6.7.1 Table 1: Session Connection Phase

* PREDICATES FOR TABLE 1

P1 : ((no resources available) or (C.init-cap = R)
or (C.cap-mode=sender-only and I.F_CONNECT_RQ.mode=receiver-only)
or (C.cap-mode=receiv-only and I.F_CONNECT_RQ.mode=sender-only)

P2 : Negotiation of:

- buffer size,
- restart,
- compression,
- mode,
- special-logic,

- credit-value is OK.

P3 : (C.init-cap = I)

P4 : mode in SSID not compatible with C.cap-mode

*** ACTIONS FOR TABLE 1**

((1)) : set V.mode = f(C.cap-mode,I.F_CONNECT_RQ.mode)
 set V.PSW, V.id, V.restart = f(I.F_CONNECT_RQ)
 set address in O.N_CONNECT_RQ
 set V.buff-size = C.max-buf-size
 set V.compression = C.cap-compression

((2)) : start inactivity timer

((3)) : set parameter in O.SSID = f(local variables)

((4)) : stop timer

((5)) : set V.mode, V.restart, V.compression, V.buff-size, V.logic, V.window =
 f(SSID)

*** NOTES FOR TABLE 1**

None.

*** TABLE 1**

STATE EVENT	IDLE	I_WF_NC	I_WF_RM	I_WF_SSID	A_NC_ONL Y	A_WF_ CONRS	OTHER STATES
F_CONNECT_RQ	A1	X	X	X	X	X	X
N_CON_CF	X	A3	X	X	X	X	X
SSRM	X	X	A8	X	X	X	X
SSID	X	X	X	A4	A5	A6	A6
N_CON_IND	A2	X	X	X	X	X	X
F_CONNECT_RS	X	UE	UE	UE	UE	A7	UE
ESID	X	X	X	A6	X	X	X

<p>(A1)</p> <p>P1:</p> <p style="padding-left: 20px;">F_ABORT_IND IDLE;</p> <p style="padding-left: 40px;">not P1</p> <p>((1))</p> <p style="padding-left: 20px;">N_CON_RQ I_WF_NC;</p>	<p>(A7)</p> <p>P2:</p> <p style="padding-left: 20px;">((4,2,5)) SSID IDLELI;</p> <p style="padding-left: 40px;">not P2:</p> <p style="padding-left: 20px;">((4,2)) ESID(R=10) F_ABORT_IND(R,AO=L) WF_NDISC;</p>
<p>(A2)</p> <p>P3:</p> <p style="padding-left: 20px;">N_DISC_RQ IDLE;</p> <p style="padding-left: 40px;">not P3:</p> <p>((2))</p> <p style="padding-left: 20px;">N_CON_RS SSRM A_NC_ONLY;</p>	<p>(A8)</p> <p>((4,2,3))</p> <p>SSID I_WF_SSID</p>
<p>(A3)</p> <p>((2))</p> <p style="padding-left: 20px;">I_WF_RM;</p>	
<p>(A4)</p> <p>P2:</p> <p style="padding-left: 20px;">((4,2,5)) F_CONNECT_CF IDLESP;</p> <p style="padding-left: 40px;">not P2:</p> <p style="padding-left: 20px;">((4,2)) ESID(R=10) F_ABORT_IND(R,AO=L) WF_NDISC;</p>	
<p>(A5)</p> <p>P4:</p> <p style="padding-left: 20px;">((4)) N_DISC_RQ IDLE;</p> <p style="padding-left: 40px;">not P4:</p> <p style="padding-left: 20px;">F_CONNECT_IND A_WF_CONRS;</p>	
<p>(A6)</p> <p>F_ABORT_IND N_DISC_RQ IDLE;</p>	

6.7.2 Table 2: error cases and abort

* PREDICATES FOR TABLE 2

None.

* ACTIONS FOR TABLE 2

None.

* NOTES FOR TABLE 2

None.

* TABLE 2

STATE \ EVENT	IDLE	I_WF_NC	WF_NDISC	OTHER STATES
TIME-OUT	X	X	N_DISC_RQ IDLE;	F_ABORT_IND N_DISC_RQ IDLE;
F_ABORT_RQ	X	N_DISC_RQ IDLE;	X	STOP TIMER N_DISC_RQ IDLE;
N_RST_IND	X	X	N_DISC_RQ IDLE;	STOP TIMER N_DISC_RQ F_ABORT_IND IDLE;
N_DISC_IND	X	F_ABORT_IND IDLE;	STOP TIMER IDLE;	STOP TIMER F_ABORT_IND IDLE;
Invalid or Unknown PDU	X	X	WF_NDISC;	STOP TIMER START TIMER ESID (R=01) F_ABORT_IND (R,AO=L) WF_NDISC;

6.7.3 Table 3: speaker part 1*** PREDICATES FOR TABLE 3**

- P1 : (I.F_START_FILE_RQ.restart-pos > 0)
/* this is a restart */
and ((V.restart = no) or (V.mode=receiver-only))
/* restart not in use for this session */
- P2 : (I.SFPA.restart-pos > V.restart-pos)
/* restart in SFPA greater than restart sent in SFID:
this is a protocol error */
- P3 : CDTCE - 1 = 0
/* credit is exhausted */
- P4 : no special logic is in use

*** ACTIONS FOR TABLE 3**

- ((1)) : Stop inactivity timer
- ((2)) : Start inactivity timer
- ((3)) : Build an EERP F_PDU using the parameter of F_EERP_RQ
- ((4)) : Store F_EERP_RQ into V.req-buf.
- ((5)) : Build SFID F_PDU using the parameters of F_START_FILE_RQ V.restart-pos = .
| F_START_FILE_RQ.restart-pos
- ((6)) : Store F_START_FILE_RQ into V.req-buf
- ((7)) : Set parameters of F_START_FILE_CF(+) with I.SFPA
- ((8)) : Set parameters of F_START_FILE_CF(-) with I.SFNA
- ((9)) : Build EERP using the parameters of F_EERP_RQ stored in V.req-buf.
- ((10)) : Build SFID using the parameter of F_START_FILE_RQ stored in V.req-buf; set V.restart-
pos
- ((11)) : Build and send an exchange buffer
- ((12)) : Set CDTCE = V.window
/* CE is the counter of emission credit */
- ((13)) : Decrease by one the CDTCE
/* CE is the counter of emission credit */
- ((14)) : Activate CRC-calculus function, add special logic header to the exchange buffer
- ((15)) : Build an NERP F_PDU using the parameter of F_NERP_RQ
- ((16)) : Store F_NERP_RQ into V.req-buf.

((17)) : Build NERP using the parameters of F_NERP_RQ stored in V.req-buf.

* NOTES FOR TABLE 3

(1) When the FTP is in the OPOWFC state (i.e. cannot send data because it is waiting for credit), there must be "a way" which the FTP can use in order to:

- prevent its user from sending F_DATA_RQ
- signal the user the possibility to re-begin sending F_DATA_RQ (after reception of command CDT)

This "way" is not described here. Its definition will be left to every local implementation; so it is a matter of a local decision as the behaviour in case of user error (UE).

(2) The choice to accept this "Request/event" while in this state is a matter of local implementation. The ODETTE state tables are based on the assumption that this event cannot occur in this state and would be considered as a user error (UE).

(3) It is a local matter to make the user aware that since the RTR is received, the protocol machine is now ready to accept the next request.

* TABLE 3

STATE \ EVENT	IDLE SP	IDLE SPCD	WF RTR	WFCD	ERST WFCD	NRST WFCD	SFST WFCD	CDST WFCD	OPOP	OPO	OPO WFC	WF_N DISC	OTHER STATES
F_EERP_RQ	A1	A1	UE (2)	A6	UE (2)	UE (2)	UE	UE	UE	UE	UE	UE	UE
F_NERP_RQ	A20	A20	UE (2)	A21	UE (2)	UE (2)	UE	UE	UE	UE	UE	UE	UE
F_START_RQ	A2	A2	UE (2)	A7	UE (2)	UE (2)	UE	UE	UE	UE	UE	X	UE
SFPA	A3	A3	A3	A3	A3	A3	A3	A3	A11	A3	A3	WF_N DISC	A3
SFNA	A3	A3	A3	A3	A3	A3	A3	A3	A12	A3	A3	WF_N DISC	A3
CD	A3	A3	A3	A8	A19	A21	A9	A10	A3	A3	A3	WF_N DISC	A3
F_DATA_RQ	UE	UE	UE	UE	UE	UE	UE	UE	UE	A13	A14	WF_N DISC	UE
CDT	A3	A3	A3	A3	A3	A3	A3	A3	A3	A16	A15	WF_N DISC	A3
F_CD_RQ	A4	UE	UE (2)	CDST WFCD	UE (2)	UE (2)	UE	UE	UE	UE	UE	X	UE
F_RELEASE_RQ R = normal	UE	A5	UE	UE	UE	UE	UE	UE	UE	UE	UE	X	UE
F_RELEASE_RQ R = error	A18	A18	A18	A18	A18	A18	A18	A18	A18	A18	A18	A18	A18
RTR	A3	A3	IDLE SP (3)	A3	A3	A3	A3	A3	A3	A3	A3	WF_N DISC	A3

<p>(A1) P4: ((1,2,3)) EERP WFRTR; not P4: ((1,2,3,14)) EERP WFRTR;</p>	<p>(A5) P4: ((1,2)) ESID(R=00) WF_NDISC; not P4: ((1,2,14)) ESID(R=00) WF_NDISC;</p>
<p>(A2) P1: UE; not P1, P4: ((1,2,5)) SFID OPOP; not(P1,P4): ((1,2,5,14)) SFID OPOP;</p>	<p>(A6) ((4)) ERSTWFCD</p> <p>(A7) P1:UE; not P1: ((6)) SFSTWFCD;</p> <p>(A8) ((1,2)) IDLESP;</p>
<p>(A3) P4: ((1,2)) ESID(R=02) F_ABORT_IND(R,AO=L) WF_NDISC; not P4: ((1,2,14)) ESID(R=02) F_ABORT_IND(R,AO=L) WF_NDISC;</p>	<p>(A9) P4: ((1,2,10)) SFID OPOP not P4: ((1,2,10,14)) SFID OPOP;</p>
<p>(A4) P4: ((1,2)) CD IDLELICD; not P4: ((1,2,14)) CD IDLELICD;</p>	<p>(A10) P4: ((1,2)) CD IDLELICD; not P4: ((1,2,14)) CD IDLELICD;</p>
<p>(A11) (P2,P4): ((1,2)) ESID(R=02) F_ABORT_IND(R,AO=L) WF_NDISC; not P2, P4: ((1,2,7,12)) F_START_FILE_CF(+) OPO; P2, not P4: ((1,2,14)) ESID(R=02) F_ABORT_IND(R,AO=L) WF_NDISC; not (P2,P4): ((1,2,7,12,14)) F_START_FILE_CF(+)</p>	<p>(A16) Protocol error P4: ((1,2)) ESID(R=02) F_ABORT_IND(R,AO=L) WF_NDISC; not P4: ((1,2,14)) ESID(R=02) F_ABORT_IND(R,AO=L) WF_NDISC;</p> <p>(A18) P4: ((1,2)) ESID(R) WF_NDISC; not P4:</p>

	OPO;		((1,2,14))
(A12)	((1,2,8))		ESID(R)
	F_START_FILE_CF(-)		WF_NDISC;
	IDLESP;	(A19)	P4:
(A13)	(P3,P4):		((1,2,9))
	((1,2,11,13))		EERP
	DATA		WFRTR;
	OPOWFC;		not P4:
	not P3, P4:		((1,2,9,14))
	((1,2,11,13))		EERP
	DATA		WFRTR;
	OPO;	(A20)	P4:
	P3, not P4:		((1,2,15))
	((1,2,11,13,14))		NERP
	DATA		WFRTR;
	OPOWFC;		not P4:
	not (P3,P4):		((1,2,15,14))
	((1,2,11,13,14))		NERP
	DATA		WFRTR;
	OPO;		
(A14)	UE (user error)	(A21)	((16))
	(1) important comment		NRSTWFCD
(A15)	OPO;	(A22)	P4:
	(1) important comment		((1,2,17))
			NERP
			WFRTR;
			not P4:
			((1,2,17,14))
			NERP
			WFRTR;

6.7.4 Table 4: speaker part 2

* PREDICATES FOR TABLE 4

P1 : (I.EFPA.CD-Request = yes) and
(V.Mode = Both)

P2 : No special logic in use

* ACTIONS FOR TABLE 4

((1)) : stop inactivity timer

((2)) : start inactivity timer

((3)) : O.F_CLOSE_FILE_CF(+).speaker=no

((4)) : O.F_CLOSE_FILE_CF(+).speaker=yes

((5)) : set parameters of EFID using those of F_CLOSE_FILE_RQ

((6)) : set parameters of F_CLOSE_FILE_CF(-) with those of EFNA

((7)) : set CDTCE = 0

((8)) : activate CRC-calculus function and, add special logic header to the exchange buffer

* NOTES FOR TABLE 4

(1) In order to respect the "half duplex" property of the ODETTE-FTP it is forbidden to send EFID while in the OPOWFC state. EFID can be sent only in the OPO state.

The FTP implementation must have a "way" to avoid sending EFID (or receiving F_CLOSE_FILE_RQ) while in the OPOWFC state. This way is not described here; its definition will be left to every local implementation.

* TABLE 4

STATE \ EVENT	OPO	OPOWFC	CLOP
F_CLOSE_FILE_RQ	A1	Note (1)	UE
EFPA	A2	A2	A2
EFNA	A2	A2	A4

(A1) P2:
 ((1,2,5,7))
 EFID
 CLOP;
 not P2:
 ((1,2,5,7,8))
 EFID
 CLOP;

(A2) ((1,2))
 ESID(R=02)
 F_ABORT_IND(R,AO=L)
 WF_NDISC;

(A3) (P1,P2):
 ((1,2,3))
 F_CLOSE_FILE_CF(+) SP=no
 CD
 IDLELI;
 P1, not P2:
 ((1,2,3,8))
 F_CLOSE_FILE_CF(+) SP=no
 CD
 IDLELI;

```
not P1:  
  ((1,2,4))  
  F_CLOSE_FILE_CF(+) SP=yes  
  IDLESP;
```

```
(A4)      ((1,2,6))  
          F_CLOSE_FILE_CF(-)  
          IDLESP;
```

6.7.5 Table 5: listener* PREDICATES FOR TABLE 5

P1 : (I.SFID.restart-pos > 0) AND (V.restart=no) /*invalid SFID*/

P2 : positive response

P3 : (I.F_CLOSE_FILE_RS(+).speaker=yes)

P4 : (I.F_START_FILE_RS(+).restart-pos > V.restart)

P5 : special logic is used

P6 : CDTCR - 1 < 0
/* protocol error because the speaker sends data
without available credit */

P7 : CDTCR - 1 = 0 /* the sender will need credit */
and the FTP state of buffers permits sending new credit and the FTP-user is
capable to receive more data

P8 : The calculus of the received CRC indicates an error

* ACTIONS FOR TABLE 5

((1)) : stop inactivity timer

((2)) : start inactivity timer

((3)) : set parameters in F_START_FILE_IND using I.SFID
V.restart-pos=I.SFID.restart-pos

((4)) : set parameters of F_EERP_IND using those of I.EERP

((5)) : add special logic header to the command to be sent to the speaker

((6)) : suppress the special logic header from the data buffer before giving it to the
user

((7)) : CDTCR = CDTCR - 1 /* decrease CDTCR by one */

((8)) : CDTCR = V.window

((9)) : Generally, the FTP-listener on receipt of an EERP or NERP will send
immediately an RTR command. If the user capacity does not permit reception
of a new EERP or NERP, then delayed RTR will take place.

((10)) : set parameters of F_NERP_IND using those of I.NERP

* NOTES FOR TABLE 5

(1) Flow control in case of reception

The FTP-Listener will send new credit to the speaker; this operation will be:

- Function of the user's (listener) capacity for reception of data.
- Function of the number of buffers available for the FTP itself
- Function of the speaker remaining credit (remaining credit must be equal to zero)

* TABLE 5

STATE EVENT	IDLELI	IDLELICD	OPIP	OPI	CLIP
SFID	A1	A1	A2	A2	A2
DATA	A2	A2	A2	A9	A2
EFID	A2	A2	A2	A10	A2
F_START_ FILE_RS	UE	UE	A8	UE	UE
F_CLOSE_ FILE_RS	UE	UE	UE	UE	A11
CD	A3	A2	A2	A2	A2
ESID R = Normal	A4	A6	A4	A4	A4
ESID R = Error	A4	A4	A4	A4	A4
EERP	A5	A7	A2	A2	A2
NERP	A12	A12	A2	A2	A2

(A1)	P1: ((1,2)) ESID(R=02) F_ABORT_IND(R,AO=L) WF_NDISC; not P1: ((1,2,3)) F_START_FILE_IND OPIP;	(A8)	P4: UE; P2 and not (P4,P5): ((1,2,8)) SFPA OPI; not (P2,P4,P5): ((1,2)) SFNA IDLELI; (P2,P5), not P4: ((1,2,5,8)) SFPA OPI; not (P2,P4), P5: ((1,2,5)) SFNA IDLELI;
(A2)	((1,2)) ESID(R=02) F_ABORT_IND(R,AO=L) WF_NDISC;		
(A3)	((1,2)) F_CD_IND IDLESPCD;		
(A4)	((1)) F_ABORT_IND (R=received ESID Reason, AO=D) N_DISC_RQ IDLE;	(A9)	P6: ((1,2)) ESID(R=02) F_ABORT_IND(R,A0,=L) WF_NDISC; not (P5,P6,P7): ((1,2,7,)) (1) F_DATA_IND OPI; not (P5,P6), P7 ((1,2,8)) (1) F_DATA_IND CDT OPI; (P5,P8), not P6: ((1,2)) ESID(R=06) F_ABORT_IND(R,AO=L) WF_NDISC; not (P6,P7,P8), P5: ((1,2,6,7)) (1) F_DATA_IND OPI; (P5,P7), not (P6,P8): ((1,2,5,6,8)) (1) F_DATA_IND CDT OPI;
(A5)	((1,2,4,9)) F_EERP_IND RTR IDLELI;		
(A6)	((1)) F_RELEASE_IND N_DISC_RQ IDLE;		
(A7)	((1,2,4,9)) F_EERP_IND RTR IDLELI;		
		(A10)	((1,2)) F_CLOSE_IND CLIP;

(A11)

```

(P2,P3) not P5:
  ((1,2))
  EFPA(CD-req.)
  WFCD;
P2, not (P3, P5):
  ((1,2))
  EFPA(no CD)
  IDLELI;
not (P2,P5):
  ((1,2))
  EFNA
  IDLELI;
not (P2,P3), P5:
  ((1,2,5))
  EFNA
  IDLELI;
(P2,P5), not P3:
  ((1,2,5))
  EFPA(no CD)
  IDLELI;
not P2, P5:
  ((1,2,5))
  EFNA
  IDLELI;
(P2, P3, P5):
  ((1,2,5))
  EFPA(CD-req.)
  WFCD;

```

(A12)

```

  ((1,2,10,9))
  F_NERP_IND
  RTR
  IDLELI;

```

VII ISO 646 – CHARACTER SUBSET

					b7	0	0	0	0	1	1	1	1
					b6	0	0	1	1	0	0	1	1
					b5	0	1	0	1	0	1	0	1
						0	1	2	3	4	5	6	7
b4	b3	b2	b1					SP	0		P		
0	0	0	0	0					1	A	Q		
0	0	0	1	1					2	B	R		
0	0	1	0	2					3	C	S		
0	0	1	1	3					4	D	T		
0	1	0	0	4					5	E	U		
0	1	0	1	5					6	F	V		
0	1	1	0	6			&		7	G	W		
0	1	1	1	7					8	H	X		
1	0	0	0	8			(9	I	Y		
1	0	0	1	9)			J	Z		
1	0	1	0	10						K			
1	0	1	1	11						L			
1	1	0	0	12									
1	1	0	1	13			-			M			
1	1	1	0	14			.			N			
1	1	1	1	15			/			O			

OFTP Revision 1.3	REMARK	PAGE: 1 DATE: 93-04-15
----------------------	--------	---------------------------

HISTORY


Odette group 4.4 was started when the first draft of the Odette File Transfer Protocol (OFTP) was finished and implementations of the protocol were beginning to appear. It was the responsibility of the group to discuss items regarding the protocol arising from the implementors, to accept or reject suggestions for modifications, to make clarifications and to resolve ambiguities. The work done by group 4.4 during the first period became eventually the OFTP version 1.2 which forms the base for the current implementations of the protocol.

The Odette group 4.4 continues to work with questions regarding the protocol and it has been felt that items discussed within the group should be documented and published.

As a result of this work, you will find at the end of OFTP Revision 1.2 in appendix two documents:

- Addendum 1: CLARIFICATION ISSUES
- Addendum 2: CONSIDERATIONS FOR FUTURE DEVELOPMENT

Those documents contain a list of the items discussed so far within group 4.4 and which will be continually updated.

	WORKING GROUP 4
	ODG04GU9112

OFTP Revision 1.3	ADDENDUM 1 CLARIFICATION ISSUES	CHAP: 1 PAGE: 1 DATE: 93-04-15
----------------------	------------------------------------	--------------------------------------

1. Clarification regarding the origin and the recipient of a file

REQUESTED BY: ---

RESPONSE: The destination on the SFID is the recipient of the file. The origin on the SFID is the originator of the file. The destination on the EERP is the originator of the file. The origin on the EERP is the recipient of the file.

2. Request to use the user data filed in the X.25 call packet

REQUESTED BY: MLC Software

RESPONSE: The user data filed in the call packet is not to be used by an OFTP implementation. This is because access to the user data filed is not always possible depending on how the X.25 support is implemented by the computer manufacturer.

The X.25 facilities field may be used, this is mentioned in the OFTP specification as parameters on the N_CON_REQ.

3. The maximum record size for files of type T is not defined in the OFTP document

REQUESTED BY: Renault

RESPONSE: The maximum record size for file type 'T' is zero (0).

4. Clarification regarding the count field on the End-of-File-id (EFID) protocol data unit


REQUESTED BY: ---

RESPONSE: The count field on the EFID protocol data unit is always the total count for the file, even during restart processing.

5. Clarification regarding the logical and physical addresses used in the File Transfer Protocol

REQUESTED BY: Sweden

RESPONSE: Clarification was requested as to whether the identities present in the SFID and SSID protocol data units represent physical nodes or if they may also be logical addresses, i.e. an application. It was clarified that the identities should be physical addresses, i.e. network nodes and that was the understanding during the development of the protocol.

	WORKING GROUP 4
	ODG04GU9112

OFTP Revision 1.3	ADDENDUM 1 CLARIFICATION ISSUES	CHAP: 1 PAGE: 2 DATE: 93-04-15
----------------------	------------------------------------	--------------------------------------

6. Clarification regarding space as embedded character in the destination/origin field in the OFTP

REQUESTED BY: ---

RESPONSE: Space is normally not allowed as an embedded character, however, for the destination and origin fields, the OFTP refers to the ISO standard IS6523 and if embedded spaces are allowed there, they must be allowed in the destination and origin fields as well.

7. Request for permission to change the session disconnect procedure

REQUESTED BY: GFI

RESPONSE: GFI have asked for a change as regards the disconnecting procedure in the protocol. Due to their product implementation they are not able to disconnect after receiving an ESID protocol data unit, instead they ask for permission to respond to the ESID by sending an ESID and having the partner disconnect the line.

This request was rejected by group 4.4.

8. Session disconnect procedure

REQUESTED BY: IBM

RESPONSE: IBM has reported a problem with their CFT/VM product. When they receive a negative response (SFNA) on a file transfer request (SFID), they disconnect without sending ESID. As general practice, this is not accepted by the group.

Consider the following scenario: a file is to be transferred in each direction, first from the IBM site and then from the partner site. If the file request from the IBM site fails (an SFNA is received by the IBM product), the line connection will be terminated and there will be no chance to send the file from the other site. However, situations can still occur when an ESID cannot be sent (due to abend etc.).

OFTP Revision 1.3	ADDENDUM 1 CLARIFICATION ISSUES	CHAP: 1 PAGE: 3 DATE: 93-04-15
----------------------	------------------------------------	--------------------------------------

9. ESID error code 3

REQUESTED BY: VW

RESPONSE: In the description for the ESID error code 3 there is a printing error. The sentence 'An SFID has been received..' should read 'An SSID has been received..'. Also, the wording 'user code' should be changed into 'identification code'.

10. XNUA sub-address in France

REQUESTED BY: Sweden

RESPONSE: Some companies seem to use the sub-address in such a way that the XNUA contains more than 14 characters. This problem seems to appear with companies which does not respect the X121 standard, because of national use of a size longer than allowed in international communication. Thus the WG4.4 group requests adherence to the the X121 standard.

11. SFNA error code

REQUESTED BY: Sweden

RESPONSE: The clarification for invalid file name in the SFNA error code is:

Invalid file name error code may be the consequence of a problem in the mapping of the virtual file on to the real file. Such problems cannot be resolved immediately in some cases. Thus, we recommend that in order to avoid a re-transmit of the file in the same session, the monitor re-transmits the relevant file in a subsequent session when receiving an SFNA with ANSWER RETRY = YES.

12. Use of SSID user data filed for binary code

REQUESTED BY: GEIS

RESPONSE: The request to use the SSID user data field for transfer of binary code is refused due to compatibility problems of existing products.

OFTP Revision 1.3	ADDENDUM 1 CLARIFICATION ISSUES	CHAP: 1 PAGE: 4 DATE: 93-04-15
----------------------	------------------------------------	--------------------------------------

13. Use of different credit numbers

REQUESTED BY: ---

RESPONSE: After study the group recommends not allowing different credit numbers depending on the direction of the transmission, because of the problem for an OFTP partner to protect itself by the credit number and not protocol units in the OFTP are defined for such kind of double negotiation. Furthermore, it seems to be better to allocate a unique credit number in the same session for performance management.

14. Restart position for U-files

REQUESTED BY: ---

RESPONSE: Restart position of unstructured (U) files are negotiated as multiples of 1K octets. 1K octets is equivalent to 1024 octets.

OFTP Revision 1.3	ADDENDUM 1 CLARIFICATION ISSUES	CHAP: 1 PAGE: 5 DATE: 95-11-30
--------------------------	--	--

15. Protocol release level

REQUESTED BY: Sweden

Response: Protocol release level should be set to 2, for Revision 1.3.
For Revision 1.2, it should be set to 1.

16. Possibility to change password automatically, using the protocol

REQUESTED BY: Ford and EDS

Response: The OFTP User field can be used for a change password mechanism. The use must be on a bilateral agreement and is not a part of the protocol itself. A suggested mechanism is presented below. ESID error codes in the range 50-70 may be used for user specific codes.

OFTP Password Change Mechanism:

An OFTP network node normally sends an SSID with its EDI code and associated password in the SSID_PASSWORD field. If the remote determines that the password is invalid, then the remote sends an error coded ESID to reject the call.

The password change mechanism requires that the side that wishes to change the password send the new password in the SSID password field and place the current password in the USERDATA field.

Consequently, the recipient of a password change request is able to determine such a request and also be able to verify the remote trading partner. The recipient of a valid password change should update their local user directory with the new password and continue with the session normally. The requester of the password change request, upon determining that the OFTP protocol is continuing without receipt of an error coded ESID can then update the requester's user directory with the new password.


Mechanism Pseudo-code:

```
// Standard ESID Error Code

#define ESID_OK 0
#define ESID_COMMAND_NOT_RECOGNISED 1
#define ESID_PROTOCOL_VIOLATION 2
#define ESID_USERCODE_NOT_KNOWN 3
#define ESID_INVALID_PASSWORD 4
#define ESID_LOCAL_SITE_EMERGENCY_CLOSE 5
#define ESID_INVALID_DATA 6
#define ESID_NSDU_SIZE_ERROR 7
#define ESID_RESOURCES_NOT_AVAILABLE 8
#define ESID_TIMEOUT 9
#define ESID_MODE_OR_CAPABILITIES 10

// Extensions to the ESID Error Codes

#define ESID_PASSORD_EXPIRED 50
```

	WORKING GROUP 4
	ODG04GU9112

OFTP Revision 1.3	ADDENDUM 1 CLARIFICATION ISSUES	CHAP: 1 PAGE: 6 DATE: 95-11-30
--------------------------	--	--

```

#define ESID_PASSORD_VALIDATION          51
#define ESID_PASSORD_BAD_FORMAT         52
#define ESID_PASSORD_TOO_SHORT         53
#define ESID_PASSORD_HISTORY           54
#define ESID_PASSORD_COMMON_NAME       55

#define ESID_ACCESS_REVOKED            56
#define ESID_ACCESS_TIME_INVALID       57
#define ESID_INVALID_NETWORK_ORIGIN    58

// The original catch-all

#define ESID_UNSPECIFIED                99

// PASSWORD CHECKING

// REMOTE_USER_PASSWORD
// is the remote user's password that is kept in this system's
// user directory

IF (SSID_PASSWORD != REMOTE_USER_PASSWORD) {
    // Standard password check has failed, so check if
    // password is in the USER field

    IF (SSID_USERDATA == REMOTE_USER_PASSWORD) {
        // Check that new password is valid

        IF (GOOD_PASSWORD(SSID_PASSWORD) == TRUE) {
            UPDATE_USER_PROFILE_WITH_NEW_PASSWORD(SSID_PAS
            SWORD);

            BREAKS;
        }


        // New password is invalid, so reject change

        ELSE {
            RETURN(ESID_PASSWORD_XXXXXX);
        }
    }

    // No valid password found anywhere, so reject

    ELSE{
        RETURN(ESID_INVALID_PASSWORD);
    }
}

```

	WORKING GROUP 4
	ODG04GU9112

OFTP Revision 1.3	ADDENDUM 1 CLARIFICATION ISSUES	CHAP: 1 PAGE: 7 DATE: 95-11-30
--------------------------	--	--

```

// Valid password (the normal situation)
// Check for expiry.

ELSE {

    IF (PASSWORD_EXPIRED() == TRUE) {

        RETURN(ESID_PASSWORD_EXPIRED);

    }

    ELSE {

        BREAK;

    }

}

PASSWORD_OK:

// PASSWORD CHECKING COMPLETE....

```

Description of the new codes:

There are a number of ESID extension codes in order that the recipient can easily determine the reason for the password change rejection.

ESID_PASSWORD_EXPIRED

Issued by either party to indicate that the password received has expired and is no longer valid for normal communications. The expiry period is a local site issue and/or may be agreed between the two parties bilaterally.

The password, although expired may be used as part of a subsequent session where the password is changed as part of the password change mechanism.

ESID_PASSWORD_VALIDATION

The new password has been rejected because it failed local site validation rules. The rules will typically be:

The password is composed of the same character repeated e.g. 'XXXXXXXX'.

ESID_PASSWORD_COMMON_NAME


The password is a common name, like JANE, SUSAN

ESID_PASSWORD_BAD_FORMAT

The password contained invalid characters and/or embedded spaces. Invalid characters could be composed from the national specific characters or lower case characters.

ESID_PASSWORD_HISTORY

The local site remembers the last 'n' passwords and disallows their use again. This is used to stop people rotating passwords,

	WORKING GROUP 4
	ODG04GU9112

OFTP Revision 1.3	ADDENDUM 1 CLARIFICATION ISSUES	CHAP: 1 PAGE: 8 DATE: 95-11-30
--------------------------	--	--

e.g. TOM-<RICHARD->TOM-<RICHARD->. The number of passwords remembered is a local site issue.

ESID_PASSWORD_TOO_SHORT

The length of the password is too short. A local site may require passwords to be of a minimum length. This is to stop trivial passwords such as 'X' and 'Y'.

ESID_ACCESS_REVOKED

Access to the OFTP system has been revoked. The OFTP code is known to the system and the password is valid, however access has been suspended. This situation could occur for various administrative reasons:

The OFTP code has been defined but has not been given live status.

The user hasn't paid the bill and has been cut off.


A site determined limit of too many incorrect access attempts has been reached e.g. a user has tried to gain access with an invalid password and has been rejected more than 'n' times. This is to stop users trying to break security by trying a range of passwords in the hope of gaining access.

ESID_ACCESS_TIME_INVALID

Some sites may wish to limit access by a particular user either to a particular day and / or a time of day. This may be to reduce communication costs or to provide enforced load balancing upon the computer system. If access is attempted outside of this bilaterally agreed time period, then access is denied.

ESID_INVALID_NETWORK_ORIGIN

The originator's X.25 NUA specified in the call request packet and / or the CLI(Calling Line identifier) in an ISDN call setup request are invalid. Some sites may wish to have much enhanced security by ensuring that a user can call from a number of predetermined network nodes.

	WORKING GROUP 4
	ODG04GU9112

OFTP Revision 1.3	ADDENDUM 2 CONSIDERATIONS FOR FUTURE DEVELOPMENT	CHAP: 2 PAGE: 1 DATE: 93-04-15
----------------------	--	--------------------------------------

1. The Odette File Transfer Protocol should provide a priority and selection mechanism in order to be able to choose the file(s) to receive

REQUESTED BY: GALIA

RESPONSE: This type of mechanism was discussed during the development of the protocol but it was decided not to include these features in the any version of the protocol to date.

2. The Odette File Transfer Protocol should provide an operator messaging mechanism

REQUESTED BY: ---

RESPONSE: This was discussed during the development of the protocol but it was decided not to include an operator messaging facility in the any version of the protocol to date.

3. The Odette File Transfer Protocol should provide for better compression methods

REQUESTED BY: ---

RESPONSE: This has been requested several times but so far it has been decided that this will not be done in the current version of the protocol.

4. The Odette File Transfer Protocol should provide a new protocol data unit to be used for sending intermediate status for files that are transferred across clearing houses

REQUESTED BY: ---

RESPONSE: When files are sent between clearing centres, there is a need for status information to be send between the centres. A proposed new protocol data unit, called Intermediate Status (EEIS) has been received from UK. The decision was to put the item under consideration for a future protocol version.

OFTP Revision 1.3	ADDENDUM 2 CONSIDERATIONS FOR FUTURE DEVELOPMENT	CHAP: 2 PAGE: 2 DATE: 93-04-15
----------------------	--	--------------------------------------

5. Lack of consistency in the special logic extension of the ODette File Transfer Protocol

REQUESTED BY: Geisco

RESPONSE: Because of the special logic support, the SSRM and SSID protocol units have a Carriage Return as ending delimiter. This ending delimiter should also be present in the ESID protocol unit for consistency in the case when a session is rejected and the SSID is responded to by an ESID.

Example:

```

SSRM /CR
----->

SSID /CR
<-----

ESID (should be a carriage return here)
----->

```

The item is accepted and will be corrected in the next version of the protocol.

6. Request for additional fields on the End-to-End-Response (EERP) protocol data unit

REQUESTED BY: ---

RESPONSE: Two extra fields are requested on the EERP protocol data unit. One is a Category and the other is a Reason code. The Category is 1 byte and if the value is zero, the file is accepted. If the value is 1, the file is rejected and the reason is given in the Reason field. The reason code field should be two bytes.

This item is being considered. There are standard Odette messages that can be used for returning information about why a file is rejected and it is not necessary to be able to pass this information via the OFTP.

OFTP Revision 1.3	ADDENDUM 2 CONSIDERATIONS FOR FUTURE DEVELOPMENT	CHAP: 2 PAGE: 3 DATE: 93-04-15
----------------------	--	--------------------------------------

7. Request for a character code indicator on the Start-File-ID protocol data unit

REQUESTED BY: Sweden

RESPONSE: The request is for a character code indicator on the SFID protocol data unit in order to inform the recipient of the file of the character representation used on the file to be transferred. The item is put on the list for future development and may be included in a future version of the protocol.