

ODETTE File Transfer Protocol

Status of this Memo

This memo provides information for the Internet community. It does not specify an Internet standard of any kind. Distribution of this memo is unlimited.

Abstract

This memo describes a file transfer protocol to facilitate electronic data interchange between trading partners.

The protocol, denoted the ODETTE File Transfer Protocol, supports both direct communication between installations and indirect communication via a third party clearing centre. It was developed by the Organisation for Data Exchange by Tele Transmission in Europe to facilitate communication within the European motor industry and is presented here to allow for wider use within the Internet community.

Table of Contents

1. Introduction	3
1.1 - Background	3
1.2 - Relationship to the original ODETTE Standard	3
1.3 - General Principles	3
1.4 - Structure	4
1.5 - Virtual Files	4
1.6 - Service Description	7
2. Network Service (TCP Transport Service)	7
2.1 - Introduction	7
2.2 - Service Primitives	7
2.3 - Port Assignment	9
3. File Transfer Service	9
3.1 - Model	10
3.2 - Session Setup	11
3.3 - File Transfer	13
3.4 - Session Take Down	16
3.5 - Service State Automata	19

4. Protocol Specification	22
4.1 - Overview	22
4.2 - Start Session Phase	22
4.3 - Start File Phase	23
4.4 - Data Transfer Phase	26
4.5 - End File Phase	27
4.6 - End Session Phase	27
4.7 - Problem Handling	28
5. Commands and Formats	28
5.1 - Conventions	28
5.2 - Commands	29
5.3 - Command Formats	29
5.4 - Identification Code	45
6. ODETTE-FTP Data Exchange Buffer	46
6.1 - Overview	46
6.2 - Data Exchange Buffer Format	46
6.3 - Buffer Filling Rules	47
7. Stream Transmission Buffer (TCP only)	47
7.1 - Introduction	47
7.2 - Stream Transmission Header Format	49
8. Protocol State Machine	50
8.1 - ODETTE-FTP State Machine	50
8.2 - Error Handling	50
8.3 - States	51
8.4 - Input Events	53
8.5 - Output Events	54
8.6 - Local Variables	55
8.7 - Local Constants	55
8.8 - Session Connection State Table	56
8.9 - Error and Abort State Table	58
8.10 - Speaker State Table 1	59
8.11 - Speaker State Table 2	63
8.12 - Listener State Table	65
8.13 - Example	68
9. Security Considerations	68
Appendix A Virtual File Mapping Example	69
Appendix B ISO 646 Character Subset	72
Acknowledgements	73
References	73
ODETTE Address	74
Author's Address	74

1. Introduction

1.1 Background

The ODETTE File Transfer Protocol (ODETTE-FTP) was defined in 1986 by working group four of the Organisation for Data Exchange by Tele Transmission in Europe (ODETTE) to address the electronic data interchange (EDI) requirements of the European automotive industry. It was designed in the spirit of the Open System Interconnection (OSI) model utilising the Network Service provided by the CCITT X25 recommendation.

Over the last ten years ODETTE-FTP has been widely deployed on systems of all sizes from personal computers to large mainframes. As a result of the wide scale deployment of internet technology and the trend towards global business practices, ODETTE has decided to extend the scope of it's file transfer protocol to allow the use of TCP/IP and to make the protocol available to the Internet community.

This memo describes the ODETTE-FTP protocol using the Transmission Control Protocol for it's network service.

1.2 Relationship to the original ODETTE Standard

This memo is an interpretation of version 1.3 of the ODETTE File Transfer Protocol [OFTP]. In the event of any ambiguity between this document and the original ODETTE-FTP, the original shall take precedence.

For ODETTE-FTP on TCP/IP the following sections have been added with respect to the original document.

- Section 2 - Network Service
- Section 7 - Stream Transmission Buffer
- Appendix A - Virtual File mapping example

1.3 General Principles

The aim of the ODETTE-FTP is to facilitate the transmission of a file between one or more locations in a way that is independent of the data communication network, system hardware and software environment.

In designing and specifying the protocol, the following factors were considered.

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).

1.4 Structure

ODETTE-FTP is modelled on the OSI reference model. It is designed to use the Network Service provided by level 3 of the model and provide a File Service to the users. Thus the protocol spans levels 4 to 7 of model.

The description of the ODETTE-FTP contained in this memo is closely related to the original 'X.25' specification of the protocol and in the spirit of the OSI model describes:

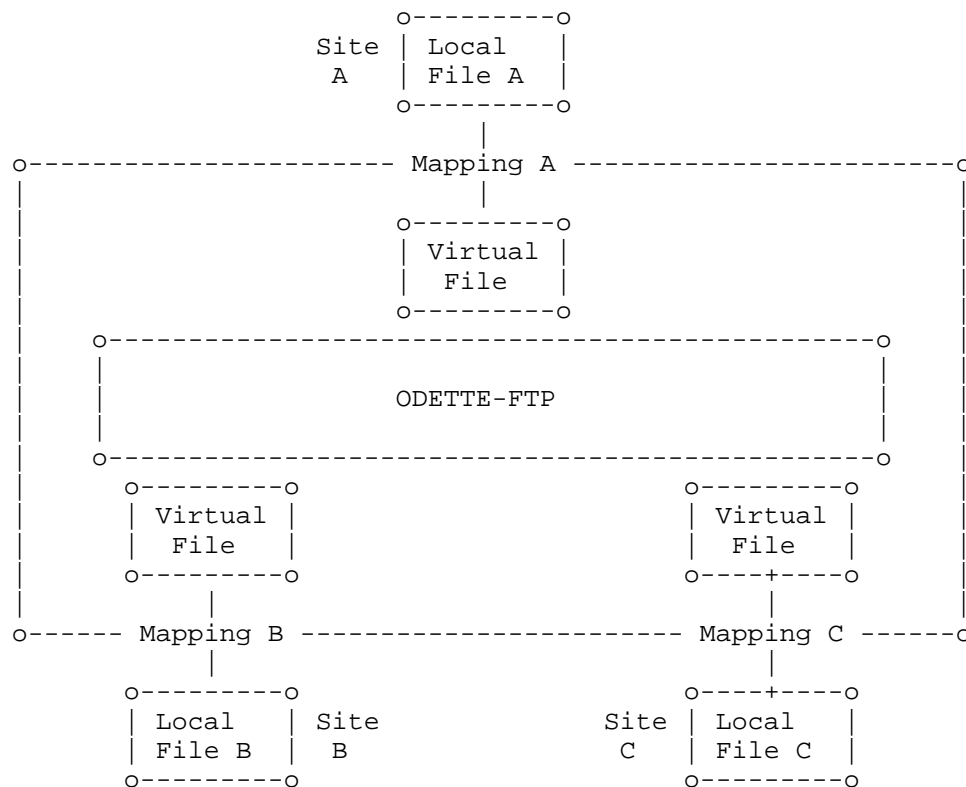
1. A File Service provided to a user monitor.
2. A protocol for the exchange of information between peer ODETTE-FTP entities.

A major consideration in adapting the protocol to use the Transmission Control Protocol (TCP) was the desire to make no changes to the existing protocol by adding the functionality required to allow implementors to support internet communication with only minor changes to existing ODETTE-FTP engines. To this end an additional header has been added to the start of each exchange buffer to allow the TCP byte stream to be broken up into the discrete exchange buffers expected by the ODETTE-FTP protocol.

1.5 Virtual Files

Information is always exchanged between ODETTE-FTP entities in a standard representation called a Virtual File. This allows data transfer without regard for the nature of the communicating systems.

The mapping of a file between a local and virtual representation will vary from system to system and is not defined here.



Time stamp (HHMMSS)

A file qualifier indicating the time the Virtual File was made available for transmission.

Date stamp (YYMMDD)

A file qualifier indicating the date the Virtual File was made available for transmission.

The Dataset Name, Date and Time attributes are assigned by a Virtual File's Originator and are used to uniquely identify the file. They must not be changed by intermediate locations.

The Date attribute represents the decade and year in a two digit field. Since the ODETTE-FTP only uses this information to identify a particular Virtual File it will continue to operate correctly in the year 2000 and beyond.

The User Monitor may use the Virtual File Date attribute in local processes involving date comparisons and calculations. Any such use falls outside the scope of this protocol and year 2000 handling is a local implementation issue.

1.5.3 Record Format

Four record formats are defined.

Fixed (F)

Each record in the file has the same length.

Variable (V)

The records in the file can have different lengths.

Unstructured (U)

The file contains a stream of data. No structure is defined.

Text File (T)

A Text File is defined as a sequence of ASCII characters, containing no control characters except CR/LF which delimits lines. A line will not have more than 2048 characters.

1.5.4 Restart

ODETTE-FTP can negotiate the restart of an interrupted Virtual File transmission. Fixed and Variable format files are restarted on record boundaries. For Unstructured and Text files the restart position is expressed as a file offset in 1K (1024 octet) blocks. The restart position is always calculated relative to the Virtual File.

1.6 Service Description

ODETTE-FTP provides a file transfer service to a user monitor and in turn uses the Internet transport layer stream service to communicate between peers.

These services are specified in this memo using service primitives grouped into four classes as follows:

Request (RQ)	An entity asks the service to do some work.
Indication (IND)	A service informs an entity of an event.
Response (RS)	An entity responds to an event.
Confirm (CF)	A service informs an entity of the response.

Services may be confirmed, using the request, indication, response and confirm primitives, or unconfirmed using just the request and indication primitives.

2. Network Service (TCP Transport Service)

2.1 Introduction

ODETTE-FTP peer entities communicate with each other via the OSI Network Service or the Transmission Control Protocol Transport Service [TCP]. This is described by service primitives representing request, indication, response and confirmation actions.

For the internet environment, the service primitives mentioned below for the Network Service have to be mapped to the respective Transport Service primitives. This section describes the network service primitives used by ODETTE-FTP and their relationship to the TCP interface. In practice the local transport service application programming interface will be used to access the TCP service.

2.2 Service Primitives

All Network primitives can be directly mapped to the respective Transport primitives when using TCP.

2.2.1 Network Connection

```

N_CON_RQ  ----->  N_CON_IND
N_CON_CF  <-----  N_CON_RS

```

This describes the setup of a connection. The requesting ODETTE-FTP peer uses the N_CON_RQ primitive to request an active OPEN of a connection to a peer ODETTE-FTP, the Responder, which has previously requested a passive OPEN. The Responder is notified of the incoming connection via N_CON_IND and accepts it with N_CON_RS. The requester is notified of the completion of it's OPEN request upon receipt of _CON_CF.

Parameters

Request	Indication	Response	Confirmation
Dest addr ----->	same	same	same

2.2.2 Network Data

```

N_DATA_RQ  ----->  N_DATA_IND

```

Data exchange is an unconfirmed service. The Requester passes data for transmission to the network service via the N_DATA_RQ primitive. The Responder is notified of the availability of data via N_DATA_IND. In practice the notification and receipt of data may be combined, such as by the return from a blocking read from the network socket.

Parameters

Request	Indication
Data ----->	same

2.2.3 Network Disconnection

```

N_DISC_RQ  ----->  N_DISC_IND

```

An ODETTE-FTP requests the termination of a connection with the N_DISC_RQ service primitive. It's peer is notified of the CLOSE by a N_DISC_IND event. It is recognised that each peer must issue a N_DISC_RQ primitive to complete the TCP symmetric close procedure.

2.2.4 Network Reset

-----> N_RST_IND

An ODETTE-FTP entity is notified of a network error by a N_RST_IND event. It should be noted that N_RST_IND would also be generated by a peer RESETTING the connection, but this is ignored here as N_RST_RQ is never sent to the Network Service by ODETTE-FTP.

2.3 Port Assignment

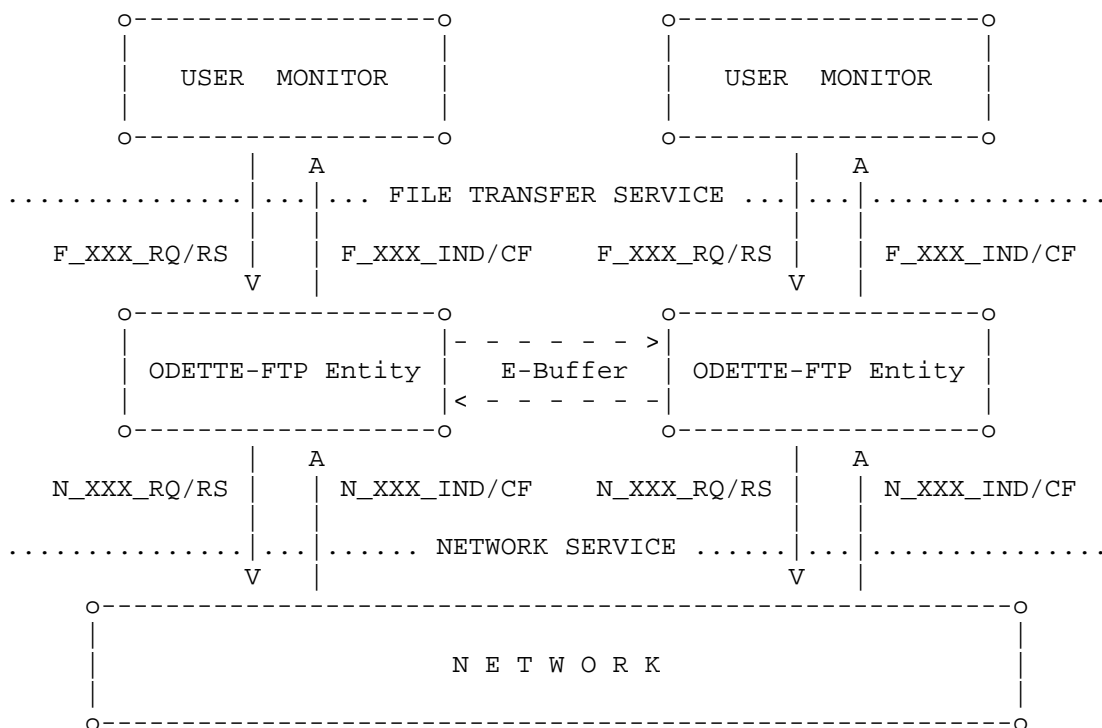
A ODETTE-FTP requester will select a suitable local port.

The responding ODETTE-FTP will listen for connections on Registered Port 3305, the service name is 'odette-ftp'.

3. File Transfer Service

The File Transfer Service describes the services offered by an ODETTE-FTP Entity to it's User Monitor. The implementation of the service primitives is a local matter.

3.1 Model



Key: E-Buffer - Exchange Buffer
 F_ - File Transfer Service Primitive
 N_ - Network Service Primitive

3.2 Session Setup

3.2.1 Session Connection Service



Parameters

Request	Indication	Response	Confirm
called-address ->	same	---	----
calling-address->	same	---	----
ID1 ----->	same	ID2 ----->	same
PSW1----->	same	PSW2 ----->	same
model ----->	mode2 ----->	mode3 ----->	same
restart1 ----->	same ----->	restart2 ----->	same

Mode

Specifies the file transfer capabilities of the entity sending or receiving a F_CONNECT primitive for the duration of the session.

Value:

Sender-Only	The entity can only send files.
Receiver-Only	The entity can only receive files.
Both	The entity can both send and receive files.

Negotiation:

Sender-Only	Not negotiable.
Receiver-Only	Not negotiable.
Both	Can be negotiated down to Sender-Only or Receiver-Only by the User Monitor or the ODETTE-FTP entity.

Request	Indication	Response	Confirm
Sender-only ---->	Receiver-only -->	Receiver-only -->	Sender-only
Receiver-only -->	Sender-only ---->	Sender-only ---->	Receiver-only
Both -----+----->	Both -----+----->	Both -----+----->	Both
	or +----->	Receiver-only -->	Sender-only
	or +----->	Sender-only ---->	Receiver-only
or +----->	Receiver-only -->	Receiver-only -->	Sender-only
or +----->	Sender-only ---->	Sender-only ---->	Receiver-only

Restart

Specifies the file transfer restart capabilities of the User Monitor.

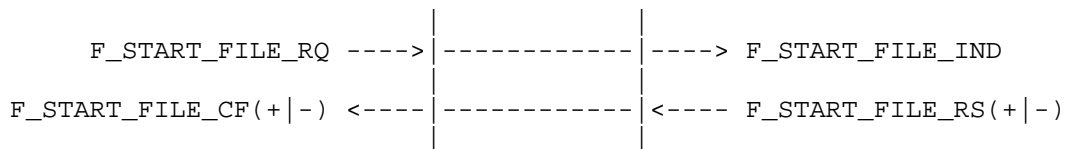
Value:

Negotiation:

Request	Indication	Response	Confirm
restart = Y ---->	restart = Y --+>	restart = Y ---->	restart = Y
	or +->	restart = N ---->	restart = N
restart = N ---->	restart = N ---->	restart = N ---->	restart = N

3.3 File Transfer

3.3.1 File Opening



Parameters:

Request	Ind.	RS(+)	CF(+)	RS(-)	CF(-)
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:

1. Retry-later has values "Y" or "N".
2. Cause is the reason for refusing the transfer (1,...,13,99).
3. Restart-pos1 not equal 0 is only valid if restart has been agreed during initial negotiation.
4. Restart-pos2 is less than or equal to restart-pos1.

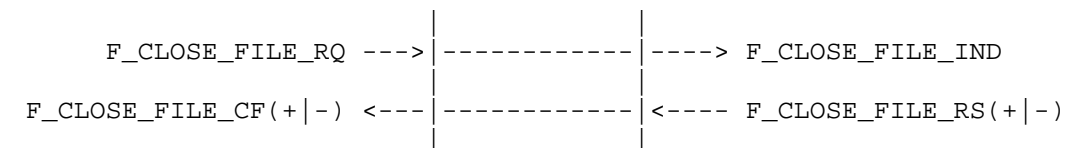
3.3.2 Data Regime



Notes:

1. The data format within a F_DATA primitive is locally defined.
2. The File Transfer service may have to provide a flow control mechanism to regulate the flow of F_DATA primitives.

3.3.3 File Closing



Parameters

Request	Ind	RS(+)	CF(+)	RS(-)	CF(-)
rec-count --->	same	----	----	----	----
unit-count -->	same	----	----	----	----
----	----	Speaker=Y --->	Speaker=N ----	----	----
----	----	Speaker=N --->	Speaker=Y ----	----	----
----	----	----	cause --->	----	same

In a positive Close File response (F_CLOSE_FILE_RS(+)) the current Speaker may either:

1. Set Speaker to "Yes" and become the Speaker.
2. Set Speaker to "No" and remain the Listener.

The File Transfer service will ensure that the setting of the speaker parameter is consistent with the capabilities of the peer user.

The turn is never exchanged in the case of a negative response or confirmation.

Only the Speaker is allowed to issue F_XXX_FILE_RQ primitives.

3.3.4 Exchanging the Turn

3.3.4.1 Initial Turn (First Speaker)

The Initiator becomes the first Speaker at the end of the Session Setup (F_CONNECT_CF received by Initiator and F_CONNECT_RS sent by Responder).

3.3.4.2 Following Turns

Rules:

1. At each unsuccessful End of File the turn is not exchanged.
2. At each successful End of File the turn is exchanged 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. The Speaker can issue a Change Direction request (F_CD_RQ) to become the Listener. The Listener receives a Change Direction indication (F_CD_IND) and becomes the 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.

3.3.5 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.



Parameters

Request	Indication

filename ----->	same
date ----->	same
time ----->	same
destination ---->	same
originator ----->	same

Relationship with Turn:

- Only the Speaker may send an End to End Response request.
- 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).

3.4 Session Take Down

3.4.1 Normal Close



Parameters

Request	Indication

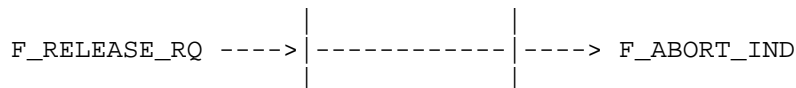
reason = normal	-----> ----

The Release service can only be initiated by the Speaker.

The Speaker can only issue a Release request (F_RELEASE_RQ) just after receiving an unsolicited Change Direction indication (F_CD_IND). This ensures that the other partner doesn't want to send any more files in this session.

Peer ODETTE-FTP entities action a normal session release by specifying Reason = Normal in an End Session (ESID) command.

3.4.2 Abnormal close



Parameters

Request	Indication

reason = error value	--> same (or equivalent)
	AO (Abort Origin) = (L)ocal or (D)istant

Abnormal session release can be initiated by either the Speaker or the Listener and also by the user or provider.

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

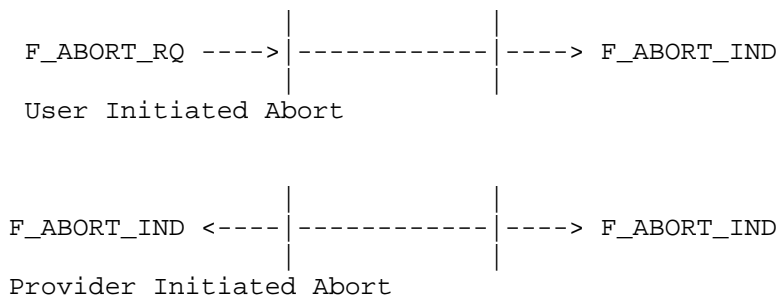
Peer ODETTE-FTP entities action an abnormal session release by specifying Reason = Error-value in an End Session (ESID) command.

The abnormal session release deals with the following types of error:

1. The service provider will initiate an abnormal release in the following cases:
 1. Protocol error, 2. Failure of the Start Session (SSID) negotiation, 3. Command not recognised, 4. Exchange buffer size error, 5. Resources not available, 6. Other unspecified abort code (with "REASON" = unspecified).
2. The User Monitor will initiate an abnormal release in the following cases:
 1. Local site emergency close down, 2. Resources not available, 3. Other unspecified abort code (with "REASON" = unspecified).

Other error types may be handled by an abort of the connection.

3.4.3 Abort



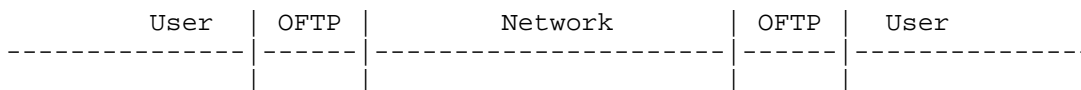
Parameters

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

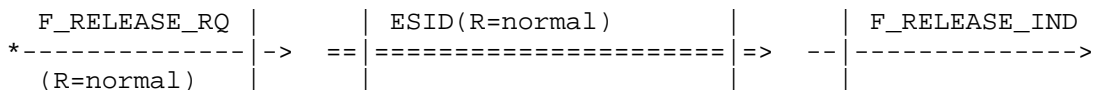
The Abort service may be invoked by either entity at any time.

The service provider may initiate an abort in case of error detection.

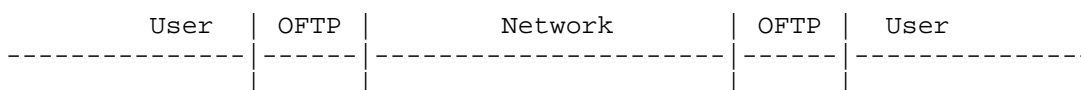
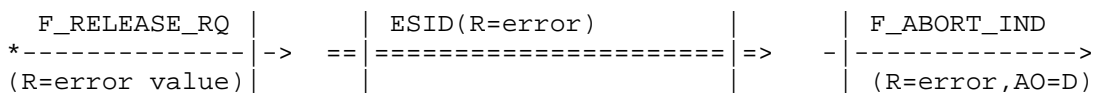
3.4.4 Explanation of Session Take Down Services



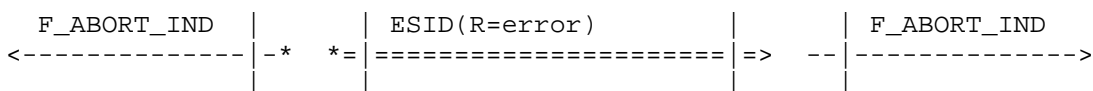
1. Normal Release



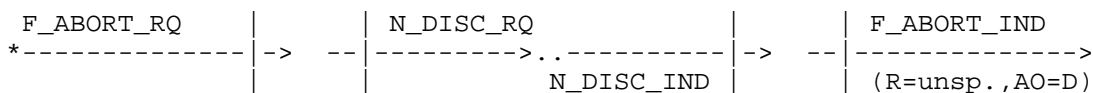
2. User Initiated Abnormal Release



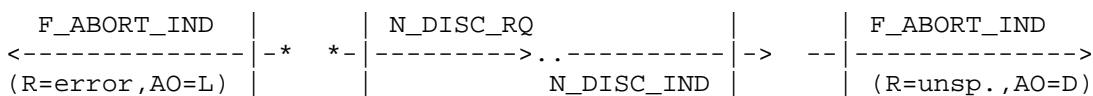
3. Provider Initiated Abnormal Release



4. User Initiated Connection Abort



5. Provider Initiated Connection Abort

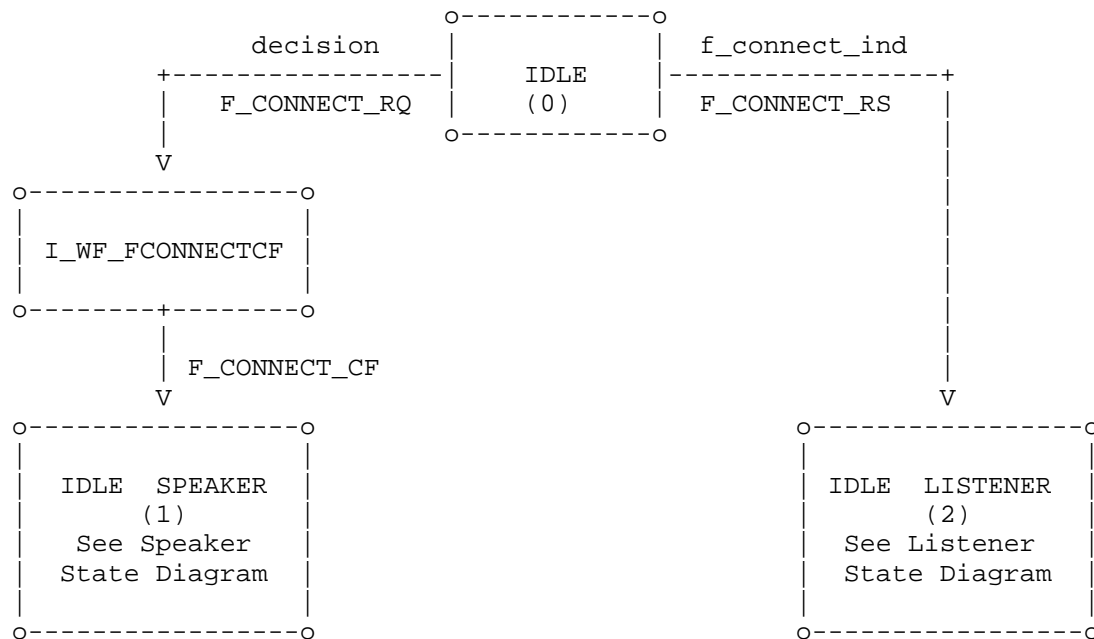


Key: * Origin of command flow
 F_ ---> File Transfer Service primitive
 N_ ---> Network Service primitive
 ==> ODETTE-FTP (OFTP) protocol message

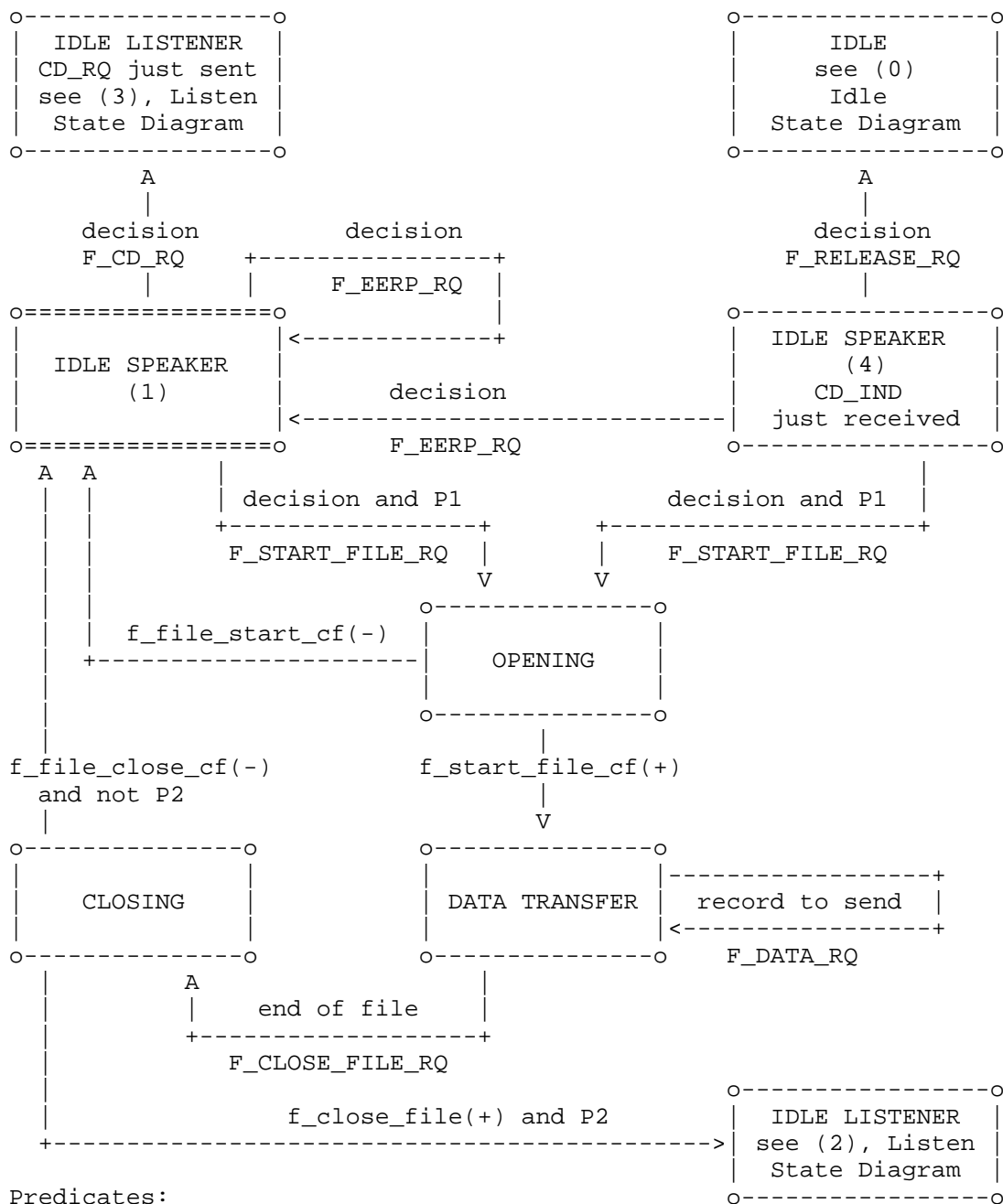
3.5 Service State Automata

This state automata defines the service as viewed by the User Monitor. Events causing a state transition are shown in lower case and the resulting action in upper case where appropriate.

3.5.1 Idle State Diagram



3.5.2 Speaker State Diagram

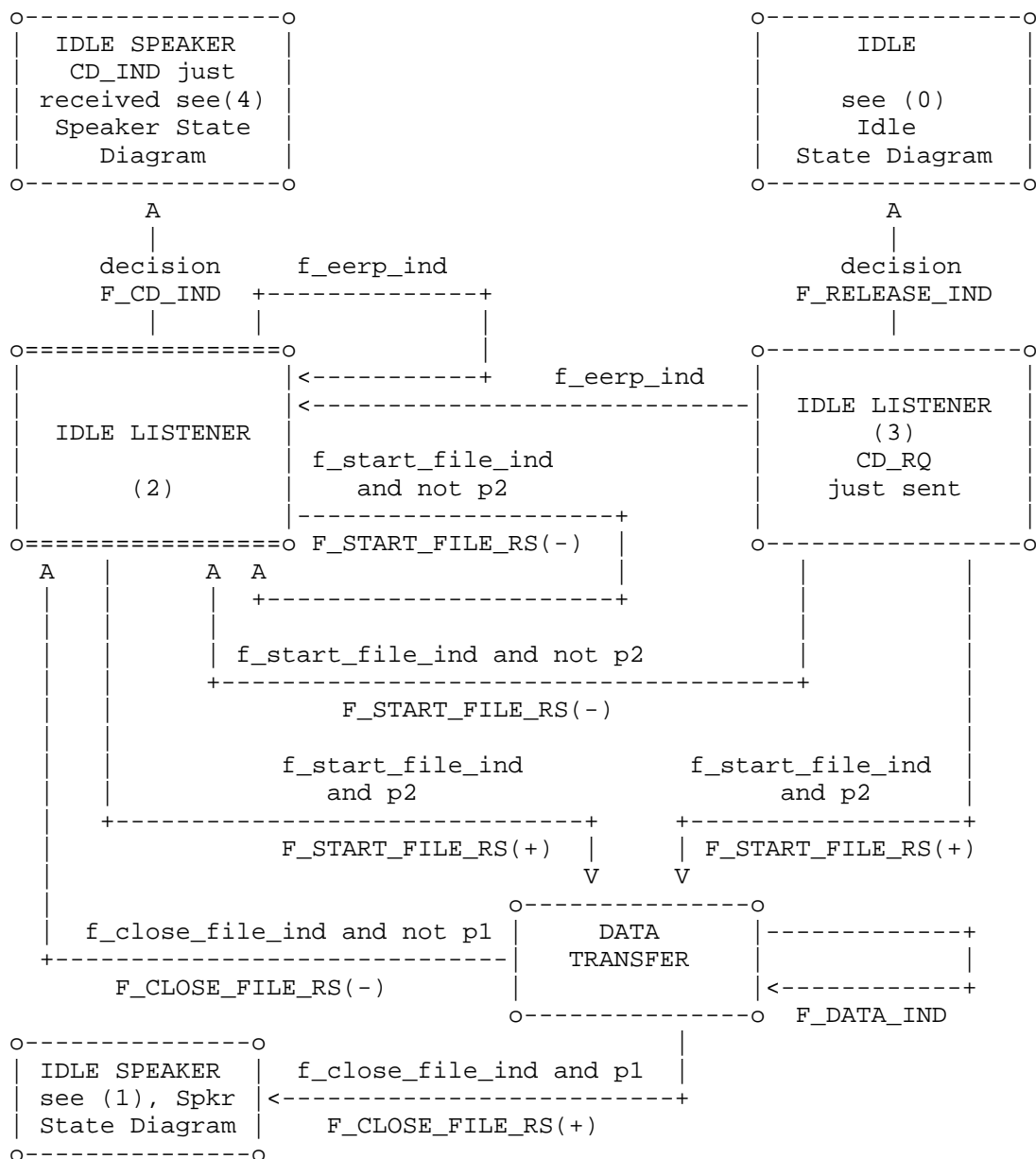


Predicates:

P1: Mode = Both or (Mode = Sender-Only)

P2: Negative confirmation or (positive confirmation, Speaker = YES)

3.5.3 Listener State Diagram



Predicates:

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(+))
```

4. Protocol Specification

4.1 Overview

The ODETTE-FTP protocol is divided into five operating phases.

- Start Session
- Start File
- Data Transfer
- End File
- End Session

After the End File phase an ODETTE-FTP entity may enter a new Start File phase or terminate the session via the End Session phase.

ODETTE-FTP peers communicate by sending and receiving messages in Exchange Buffers via the Network Service. Each Exchange Buffer contains one of the following commands.

SSRM	Start Session Ready Message
SSID	Start Session
SFID	Start File
SFPA	Start File Positive Answer
SFNA	Start File Negative Answer
DATA	Data
CDT	Set Credit
EFID	End File
EFPA	End File Positive Answer
EFNA	End File Negative Answer
ESID	End Session
CD	Change Direction
EERP	End to End Response
RTR	Ready To Receive

The remainder of this section describes the protocol flows. Section five details the command formats.

4.2 Start Session Phase

The Start Session phase is entered immediately after the network connection has been established.

4.2.1 Entity Definition

The ODETTE-FTP entity that took the initiative to establish the network connection becomes the Initiator. It's peer becomes the Responder.

4.2.2 Protocol Sequence

The first message must be sent by the Responder.

```
1. Initiator <-----SSRM -- Responder    Ready Message
              -- SSID ----->            Identification
              <----- SSID --            Identification
```

4.3 Start File Phase

4.3.1 Entity Definition

The Initiator from the Start Session phase is designated the Speaker while the Responder becomes the Listener. The roles are reversed by the Speaker sending a Change Direction command to the Listener.

4.3.2 Protocol Sequence

```
1. Speaker  -- SFID -----> Listener    Start File
            <----- SFPA --            Answer YES
```

```
2. Speaker  -- SFID -----> Listener    Start File
            <----- SFNA --            Answer NO
                Go To 1
```

Note: The User Monitor should take steps to prevent a loop situation occurring.

```
2. Speaker  -- CD -----> Listener    Change Direction
    Listener <----- EERP -- Speaker    End to End Response
            -- RTR ----->            Ready to Receive
            <----- SFID --            Start File
```

4.3.3 Restart Facilities

The Start File command includes a count allowing the restart of an interrupted transmission to be negotiated. If restart facilities are not available the restart count must be set to zero. The sender will start with the lowest record count + 1.

4.3.4 Broadcast Facilities

The destination in a Start File command can be specified as follows.

1. An explicitly defined destination.

2. A group destination that allows an intermediate location to broadcast the Virtual File to multiple destinations.

The Listener will send a negative answer to the Speaker when the destination is not known.

4.3.5 Priority

The prioritisation of files for transmission is left to the local implementation. To allow some flexibility, a change direction mechanism is available in the End File phase.

4.3.6 End To End Response (EERP)

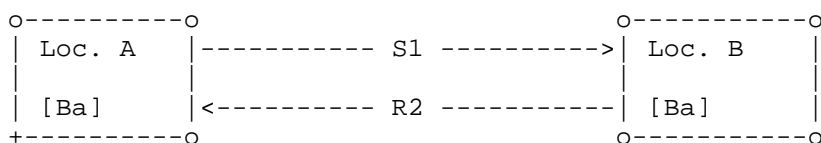
The End to End Response (EERP) command notifies the originator of a Virtual File that it has been successfully delivered to it's final destination. This allows the originator to perform house keeping tasks such as deleting copies of the delivered data.

A Response Command must be sent from the location performing the final processing or distribution of the data to the originator. The Response is mandatory and may be sent in the same or in any subsequent session.

When an intermediate location broadcasts or distributes a Virtual File it must receive a Response command from all the locations to which it forwarded the data before sending it's own Response. This ensures that the Response received by the Virtual File's originator accounts for all the destination locations. An intermediate location therefore needs to track the status of files it processes over time.

Example: Point to Point

Location A sends file Ba to Location B which will send an EERP to location A after it successfully receives the file.



Key:

S - File Transfer R - Response EERP [Ba] - File for B from A


```

Speaker  -- EERP -----> Listener    End to End Response
<----- RTR --              Ready to Receive
-- EERP ----->                    End to End Response
<----- RTR --                    Ready to Receive
-- SFID ----->                    Start File
      or
-- CD ----->                      Exchange the turn

```

After sending an EERP, the Speaker must wait for an RTR before sending any other commands.

4.4 Data Transfer Phase

Virtual File data flows from the Speaker to the Listener during the Data Transfer phase which is entered after the Start File phase.

4.4.1 Protocol Sequence

To avoid congestion at the protocol level a flow control mechanism is provided via the Credit (CDT) command.

A Credit limit is negotiated in the Start Session phase, this represents the number of Data Exchange Buffers that the Speaker may send before it is obliged to wait for a Credit command from the Listener.

The available credit is initially set to the negotiated value by the Start File positive answer, which acts as an implicit Credit command. The Speaker decreases the available credit count by one for each data buffer sent to the Listener.

When the available credit is exhausted, the Speaker must wait for a Credit command from the Listener otherwise a protocol error will occur and the session will be aborted.

The Listener should endeavour to send the Credit command without delay to prevent the Speaker blocking.

```

1. Speaker  -- SFID -----> Listener      Start File
              <----- SFPA --              Answer YES

```

2. If the Credit Value is set to 2

```

Speaker  -- Data -----> Listener  Start File
          -- Data ----->
          <----- CDT --           Set Credit
          -- Data ----->
          -- EFID ----->           End File

```

4.5 End File Phase

4.5.1 Protocol Sequence

The Speaker notifies the Listener that it has finished sending a Virtual File by sending an End File (EFID) command. The Listener replies with a positive or negative End File command and has the option to request a Change Direction command from the Speaker.

```

1. Speaker  -- EFID -----> Listener  End File
            <----- EFPA --           Answer YES

2. Speaker  -- EFID -----> Listener  End File
            <----- EFPA --           Answer YES + CD
            -- CD ----->             Change Direction
Listener <----- EERP -- Speaker  End to End Response
            ----- RTR ->           Ready to Receive
            Go to Start File Phase

3. Speaker  -- EFID -----> Listener  End File
            <----- EFNA --           Answer NO

```

4.6 End Session Phase

4.6.1 Protocol Sequence

The Speaker terminates the session by sending an End Session (ESID) command.

```

1. Speaker  -- EFID -----> Listener  End File
            <----- EFPA --           Answer YES
            -- CD ----->             Change Direction
Listener <----- ESID -- Speaker  End Session

```

4.7 Problem Handling

Error detection and handling should be done as close as possible to the problem. This aids problem determination and correction. Each layer of the reference model is responsible for its own error handling.

ODETTE-FTP can detect protocol errors through the construction of its state machine, and uses activity timers to detect session hang conditions. These mechanisms are separate from the End to End controls.

4.7.1 Protocol Errors

If a protocol error occurs the session will be terminated and application activity aborted. Both locations enter the IDLE state.

4.7.2 Timers

To protect against application and network hang conditions ODETTE-FTP uses activity timers for all situations where a response is required. The timers and actions to be taken if they expire are described in section 8, the Protocol State Machine.

4.7.3 Clearing Centres

The use of clearing centres introduces the possibility of errors occurring as a result of data processing activities within the centre. Such errors are not directly related to ODETTE-FTP or the communication network and are therefore outside the scope of this specification.

5. Commands and Formats

ODETTE-FTP entities communicate via Exchange Buffers. The Command Exchange Buffers are described below. Virtual File data is carried in Data Exchange Buffers which are described in Section 6.

5.1 Conventions

5.1.1 Representation unit:

The basic unit of information is an octet, containing eight bits.

5.1.2 Values and Characters:

The ISO 646 IRV 7-bit coded character set [ISO-646] is used to encode constants and strings within command exchange buffers.

5.2 Commands

A Command Exchange Buffer contains a single command starting at the beginning of the buffer. Commands and data are never mixed within an Exchange Buffer. Each command has a fixed length and can not be compressed.

Components:

1. Command identifier:

The first octet of an Exchange Buffer is the Command Identifier and defines the format of the buffer.

2. Parameter(s):

Command parameters are stored in fixed fields within a Command Exchange Buffer. All values are required.

5.3 Command Formats

The ODETTE-FTP commands are described below using the following definitions.

Position (Pos.)

Field offset within the command exchange buffer, relative to a zero origin.

Field

The name of the field.

Description

A description of the field.

Format

F - A field containing fixed values. All allowable values for the field are enumerated in the command definition.

V - A field with variable values within a defined range. For example the SFIDFSIZ field may contain any integer value between 0000000 and 9999999.

X(n) - An alphanumeric field of length n octets.

SSIDSDEB Exchange Buffer Size Numeric(5)

Minimum: 128
Maximum: 99999

The length, in octets, of the largest Exchange Buffer that can be accepted by the location. The length includes the command octet but does not include the Stream Transmission Header.

After negotiation the smallest size will be selected.

SSIDSR Send / Receive Capabilities Character

Value: 'S' Location can only send files.
 'R' Location can only receive files.
 'B' Location can both send and receive files.

Sending and receiving will be serialised during the session, so parallel sessions will not take place.

An error occurs if adjacent locations both specify the send or receive capability.

SSIDCMPR Compression Indication Character

Value: 'Y' The location can handle compressed data.
 'N' The location can not handle compressed data.

Compression is only used if supported by both locations. The compression mechanism is described in Section 6.2

SSIDREST Restart Indication Character

Value: 'Y' The location can handle the restart of a partially transmitted file.
 'N' The location can not restart a file.

SSIDSPEC Special Logic Indication Character

Value: 'N' Only valid value for TCP.

The Special Logic extensions are only useful in an X.25 environment and are not supported for TCP/IP.

SSIDCRED	Credit	Numeric(3)
----------	--------	------------

Maximum: 999

The number of consecutive Data Exchange Buffers sent by the Speaker before it must wait for a Credit (CDT) command from the Listener.

The credit value is only applied to Data flow in the Data Transfer phase.

The Speaker's available credit is initialised to SSIDCRED when it receives a Start File Positive Answer (SFPA) command from the Listener. It is zeroed by the End File (EFID) command.

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

Negotiation of the "credit-window-size" parameter.

```

Window Size m  -- SSID ----->
<----- SSID -- Window Size n
                  (n less or equal m)

```

Note: negotiated value will be "n".

SSIDRSV1	Reserved	String(5)
----------	----------	-----------

This field is reserved for future use.

SSIDUSER	User Data	String(8)
----------	-----------	-----------

May be used by the ODETTE-FTP in any way. If unused it should be initialised to spaces. It is expected that a bilateral agreement exists as to the meaning of the data.

SSIDCR	Carriage Return	Character
--------	-----------------	-----------

Value: Character with hex value '0D' or '8D'.

SFIDTIME Virtual File Time stamp String(6)

Format: 'HHMMSS' 6 decimal digits representing hours, minutes and seconds respectively [ISO-8601].

Time stamp assigned by the Virtual File's Originator indicating when the file was made available for transmission.

See Virtual Files - Identification (Section 1.5.2)

SFIDUSER User Data String(8)

May be used by the ODETTE-FTP in any way. If unused it should be initialised to spaces. It is expected that a bilateral agreement exists as to the meaning of the data.

SFIDDEST Destination String(25)

Format: See Identification Code (Section 5.4)

The Final Recipient of the Virtual File.

This is the location that will look into the Virtual File content and perform mapping functions. It is also the location that creates the End to End Response (EERP) command for the received file.

SFIDORIG Originator String(25)

Format: See Identification Code (Section 5.4)

Originator of the Virtual File.
It is the location that created (mapped) the data for transmission.

SFIDFMT File Format Character

Value: 'F' Fixed format binary file.
'V' Variable format binary file.
'U' Unstructured binary file.
'T' Text

Virtual File format. Used to calculate the restart position. (Section 1.5.3)

SFIDLRECL Maximum Record Size Numeric(5)

Maximum: 99999

Length in octets of the longest logical record which may be transferred to a location. Only user data is included.

If SFIDFMT is 'T' or 'U' then this attribute must be set to '00000'.

SFIDFSIZ File Size Numeric(7)

Maximum: 9999999

Space in 1K (1024 octet) blocks required at the Originator location to store the Virtual File.

This parameter is intended to provide only a good estimate of the Virtual File size.

SFIDREST Restart Position Numeric(9)

Maximum: 999999999

Virtual File restart position.

The count represents the:

- Record Number if SSIDFMT is 'F' or 'V'.
- File offset in 1K (1024 octet) blocks if SSIDFMT is 'U' or '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.

5.3.4 SFPA - Start File Positive Answer

SFPA Start File Positive Answer			
Start File Phase		Speaker <---- Listener	
Pos	Field	Description	Format
0	SFPACMD	SFPA Command, '2'	F X(1)
1	SFPAACNT	Answer Count	V 9(9)

SFPACMD Command Code Character

Value: '2' SFPA Command identifier.

SFPAACNT Answer Count Numeric(9)

The Listener must enter a count lower or equal to the restart count specified by the Speaker in the Start File (SFID) command. The count expresses the received user data. If restart facilities are not available, a count of zero must be specified.

5.3.5 SFNA - Start File Negative Answer

SFNA Start File Negative Answer			
Start File Phase		Speaker <---- Listener	
Pos	Field	Description	Format
0	SFNACMD	SFNA Command, '3'	F X(1)
1	SFNAREAS	Answer Reason	F 9(2)
3	SFNARRTR	Retry Indicator, (Y/N)	F X(1)

SFNACMD Command Code Character

Value: '3' SFNA Command identifier.

SFNAREAS Answer Reason Numeric(2)

Value: '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.
 '99' Unspecified reason.

Reason why transmission can not proceed.

SFNARTR Retry Indicator

Character

Value: 'N' Transmission should not be retried.
 'Y' The transmission may be retried latter.

This parameter is used to advise the Speaker if it should retry at a latter point in time due to a temporary condition at the Listener site, such as a lack of storage space. It should be used in conjunction with the Answer Reason code (SFNAREAS).

An invalid file name error code may be the consequence of a problem in the mapping of the Virtual File on to a real file. Such problems cannot always be resolved immediately. It is therefore recommended that when a SFNA with Retry = Y is received the User Monitor attempts to retransmit the relevant file in a subsequent session.

5.3.6 DATA - Data Exchange Buffer

O-----O			
DATA		Data Exchange Buffer	
Data Transfer Phase		Speaker ----> Listener	
Pos	Field	Description	Format
0	DATA CMD	DATA Command, 'D'	F X(1)
1	DATABUF	Data Exchange Buffer payload	V X(n)
O-----O			

DATA CMD Command Code

Character

Value: 'D' DATA Command identifier.

DATABUF Data Exchange Buffer payload

String(n)

Variable length buffer containing the data payload. The Data Exchange Buffer is described in Section 6.

5.3.7 CDT - Set Credit

CDT Set Credit			
Data Transfer Phase		Speaker <---- Listener	
Pos	Field	Description	Format
0	CDTCMD	CDT Command, 'C'	F X(1)
1	CDTRSV1	Reserved	F X(2)

CDTCMD Command Code Character

Value: 'C' CDT Command identifier.

CDTRSV1 Reserved String(2)

This field is reserved for future use.

5.3.8 EFID - End File

EFID End File			
End File Phase		Speaker ----> Listener	
Pos	Field	Description	Format
0	EFIDCMD	EFID Command, 'T'	F X(1)
1	EFIDRCNT	Record Count	V 9(9)
10	EFIDUCNT	Unit Count	V 9(12)

EFIDCMD Command Code Character

Value: 'T' EFID Command identifier.

EFIDRCNT Record Count Numeric(9)

Maximum: 999999999

For SSIDFMT 'F' or 'V' the exact record count.

For SSIDFMT 'U' or 'T' zeros.

'02' Protocol violation

An Exchange Buffer contains an invalid command for the current state of the receiver.

'03' User code not known

A Start Session (SSID) command contains an unknown or invalid Identification Code.

'04' Invalid password

A Start Session (SSID) command contained an invalid password.

'05' Local site emergency close down

The local site has entered an emergency close down mode. Communications are being forcibly terminated.

'06' Command contained invalid data

A field within a Command Exchange buffer contains invalid data.

'07' Exchange Buffer size error

The length of the Exchange Buffer as determined by the Stream Transmission Header is different to the length implied by the Command Code.

'08' Resources not available

The request for connection has been denied due to a resource shortage. The connection attempt should be retried later.

'09' Time out

'10' Mode or capabilities incompatible

'99' Unspecified Abort code

An error was detected for which no specific code is defined.

ESIDCR Carriage Return

Character

Value: Character with hex value '0D' or '8D'.

5.3.12 CD - Change Direction

CD Change Direction			
Start File Phase		Speaker ---->	Listener
End File Phase		Speaker ---->	Listener
End Session Phase		Initiator <---->	Responder
Pos	Field	Description	Format
0	CDCMD	CD Command, 'R'	F X(1)

CDCMD Command Code

Character

Value: 'R' CD Command identifier.

5.3.13 EERP - End to End Response

EERP End to End Response			
Start File Phase		Speaker ---->	Listener
End File Phase		Speaker ---->	Listener
Pos	Field	Description	Format
0	EERPCMD	EERP Command, 'E'	F X(1)
1	EERPDSN	Virtual File Dataset Name	V X(26)
27	EERPRSV1	Reserved	F X(9)
36	EERPDATE	Virtual File Date stamp, (YYMMDD)	V X(6)
42	EERPTIME	Virtual File Time stamp, (HHMMSS)	V X(6)
48	EERPUSER	User Data	V X(8)
56	EERPDEST	Destination	V X(25)
81	EERPORIG	Originator	V X(25)

EERPCMD Command Code

Character

Value: 'E' EERP Command identifier.

SIOICD	International Code Designator	String(4)
	A code forming part of the Organisation Identifier.	
SIOORG	Organisation Code	String(14)
	A code forming part of the Organisation Identifier. This field may contain the letters A to Z, the digits 0 to 9, apace and hyphen characters.	
SIOCSA	Computer Sub-Address	String(6)
	A locally assigned address which uniquely identifies a system within an organisation (defined by an Organisation Identifier).	

6. ODETTE-FTP Data Exchange Buffer

6.1 Overview

Virtual Files are transmitted by mapping the Virtual File records into Data Exchange Buffers, the maximum length of which was negotiated between the ODETTE-FTP entities via the Start Session (SSID) commands exchanged during the Start Session Phase of the protocol. The format is based on the Network Independent File Transfer Protocol [NIFTP].

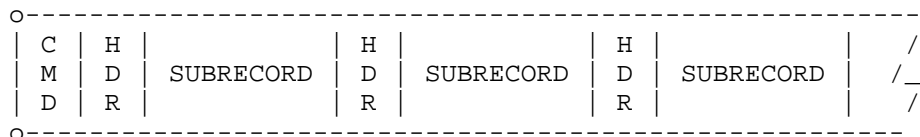
Virtual File records may be of arbitrary length. A simple compression scheme is defined for strings of repeated characters.

An example of the use of the Data Exchange Buffer can be found in Appendix A.

6.2 Data Exchange Buffer Format

For transmission of Virtual File records, data is divided into Subrecords, each of which is preceded by a one octet Subrecord Header.

The Data Exchange Buffer is made up of the initial Command character,

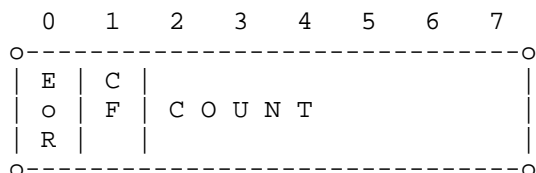


CMD

The Data Exchange Buffer Command Character, 'D'.

HDR

A one octet Subrecord Header defined as follows:



Bits

0 End of Record Flag

Set to indicate that the next subrecord is the last subrecord of the current record.

Unstructured files are transmitted as a single record, in this case the flag acts as an end of file marker.

1 Compression Flag

Set to indicate that the next subrecord is compressed.

2-7 Subrecord Count

The number of octets in the Virtual File represented by the next subrecord expressed as a binary value.

For uncompressed data this is simply the length of the subrecord.

For compressed data this is the number of times that the single octet in the following subrecord must be inserted in the Virtual File.

As six bits are available, the next subrecord may represent between 0 and 63 octets of the Virtual File.

6.3 Buffer Filling Rules

An Exchange Buffer may be any length up to the value negotiated in the Start Session exchange.

Virtual File records may be concatenated within one Exchange Buffer or split across a number of buffers.

A subrecord is never split between two Exchange Buffers. If the remaining space in the current Exchange Buffer is insufficient to contain the next 'complete' subrecord one of the following strategies should be used:

1. Truncate the Exchange Buffer, and put the complete subrecord (preceded by its header octet) in a new Exchange Buffer.
2. Split the subrecord into two, filling the remainder of the Exchange Buffer with the first new subrecord and starting a new Exchange Buffer with the second.

A record of length zero may appear anywhere in the Exchange Buffer.

A subrecord of length zero may appear anywhere in the record and/or the Exchange Buffer.

7. Stream Transmission Buffer (TCP only)

7.1 Introduction

The ODETTE-FTP was originally designed to utilise the ISO Network Service, specifically the X.25 specification. It relies on the fact that the network service will preserve the sequence and boundaries of data units transmitted through the network and that the network service will pass the length of the data unit to the receiving ODETTE-FTP. The TCP offers a stream based connection which does not provide these functions.

In order to utilise the TCP stream without disruption to the existing ODETTE-FTP a Stream Transmission Buffer (STB) is created by adding a Stream Transmission Header (STH) to the start of all Command and Data Exchange Buffers before they are passed to the TCP transport service. This allows the receiving ODETTE-FTP to recover the original Exchange Buffers.

STH - Stream Transmission Header
OEB - ODETTE-FTP Exchange Buffer

The Stream Transmission Buffer comprises of a STH and OEB.

```

O-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| STH | OEB           | STH | OEB           | STH | OEB/
O-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+

```

7.2 Stream Transmission Header Format

The Stream Transmission Header is shown below. The fields are transmitted from left to right.

```

      0               1               2               3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
|Version|  Flags  | Length                                     |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

Version

Value: 0001 (binary)

Stream Transmission Header version number.

Flags

Value: 0000 (binary)

Reserved for future use.

Length

Range: 5 - 100003 (decimal)

The length of the Stream Transmission Buffer (STH+OEB).

The smallest STB is 5 octets consisting of a 4 octet header followed by a 1 octet Exchange Buffer such as a Change Direction (CD) command.

The maximum Exchange Buffer length that can be negotiated is 99999 octets (Section 5.3.2) giving a STB length of 100003.

The length is expressed as a binary number with the most significant bit on the left.

It is expected that implementations of this protocol will follow the Internet robustness principle of being conservative in what is sent and liberal in what is accepted.

8. Protocol State Machine

8.1 ODETTE-FTP State Machine

The operation of an ODETTE-FTP entity is formally defined by the State Machine presented below. There are five State and Transition tables and for each table additional information is given in the associated Predicate and Action lists.

The response of an ODETTE-FTP entity to the receipt of an event is defined by a Transition table entry indexed by the Event/State intersection within the appropriate State table.

Each Transition table entry defines the actions taken, events generated and new state entered. Predicates may be used within a table entry to select the correct response on the basis of local information held by the entity.

A transition table contains the following fields:

Index(I)	State transition index.
Predicate	A list of predicates used to select between different possible transitions. The predicates are defined in the Predicate and Action list.
Actions	A list of actions taken by the entity. The actions are defined in the Predicate and Action list.
Events	Output events generated by the entity
Next State	The new state of the entity.

8.2 Error Handling

The receipt of an event in a given state may be invalid for three reasons.

1. The case is impossible by construction of the state automata, denoted 'X' in the State tables. For example a timer which has not been set cannot run out.
2. The event is the result of an error in the Network Service implementation, also denoted 'X' in the state tables. The Network Service implementation is considered to be correct.
3. For all other cases the event is considered to be a User Error, denoted "U" in the state tables.

The State tables define the conditions under which a User event is valid, thus preventing the generation of a protocol error by the ODETTE-FTP entity as a result of a User Monitor error. The reaction of the entity to such errors is undefined and regarded as a local implementation issue.

The State tables also allow protocol errors due to the receipt of invalid Exchange Buffers, to be detected. In such cases the reaction of the entity to the error is defined.

8.3 States

The Command Mode is strictly a Half Duplex Flip-Flop Mode.

A_NC_ONLY Responder, Network Connection opened

The Responder has sent it's Ready Message (SSRM) and is waiting for Start Session (SSID) from the Initiator.

A_WF_CONRS Responder Waiting for F_CONNECT_RS

The Responder has received the Initiator's Start Session (SSID) and is waiting for a response (F_CONNECT_RS) from it's User Monitor.

CDSTWFCD CD_RQ stored in WF_CD state

Since the User Monitor doesn't see the WF_CD state it may send a Change Direction request (F_CD_RQ) before the ODETTE-FTP receives a Change Direction (CD) command.

CLIP Close Input Pending

The Listener has received an End File (EFID) command and is waiting for the Close File response (F_CLOSE_FILE_RS) from it's User Monitor.

CLOP Close Out Pending

The Speaker has sent an End File (EFID) command and is waiting for an End File Answer (EFPA or EFNA).

ERSTWFCD End to End Response stored in WF_CD state

Since the User Monitor doesn't see the WF_CD state it may send F_EERP_RQ, before the ODETTE-FTP receives a Change Direction (CD) command.

IDLE	Connection IDLE
IDLELI	Idle Listener
IDLELICD	Idle Listener, F_CD_RQ Received The ODETTE-FTP entity has become the Listener after receiving a Change Direction request (F_CD_RQ) from the User Monitor. The receipt of an End Session (ESID) is valid in this state.
IDLESP	Idle Speaker
IDLESPCD	Idle Speaker, F_CD_IND Sent The ODETTE-FTP entity has sent a Change Direction indication (F_CD_IND) to the User Monitor. A Change Direction request (F_CD_RQ) is invalid in this state.
I_WF_NC	Initiator Waiting for Network Connection The Initiator has requested a new network connection and is waiting for a Connection confirmation (N_CON_CF) from the Network Service.
I_WF_RM	Initiator Waiting for Ready Message Before sending Start Session (SSID), the Initiator must wait for a Ready Message (SSRM) from the Responder.
I_WF_SSID	Initiator Waiting for SSID The Initiator has sent a Start Session (SSID) command and is waiting for Start Session from the Responder.
OPI	Open Input (Data Transfer Phase) The Listener is waiting for the Speaker to send a Data Exchange buffer.
OPIP	Open Input Pending The Listener has received a Start File (SFID) command and is waiting for the Start File response (F_START_FILE_RS) from it's User Monitor.

OPO	Open Out (Data Transfer Phase)
	The Speaker has received a Start File Positive Answer (SFPA) and is waiting for a Data (F_DATA_RQ) or Close File (F_CLOSE_FILE) request from it's User Monitor.
OPOP	Open Out Pending
	The Speaker has sent a Start File (SFID) command and is waiting for a Start File Answer (SFPA or SFNA).
OPOWFC	Open Out Wait for Credit
	The Speaker is waiting for a Set Credit (CDT) command before sending further Data Exchange buffers.
SFSTWFC	Start File Request stored in WF_CD state.
	Since the User Monitor doesn't see the WF_CD state it may send a Start File request (F_START_FILE_RQ) before the ODETTE-FTP receives a Change Direction (CD) command.
WF_CD	Wait for Change Direction
	The Listener wishes to become the Speaker and is waiting for a Change Direction (CD) command after sending an End File Positive Answer (EFPA) requesting change direction.
WF_RTR	Wait for Ready To Receive
	The Initiator has sent an End to End Response (EERP) command and must wait for Ready To Receive (RTR) from the Responder.
WF_NDISC	Wait for N_DISC_IND
	ODETTE-FTP has sent an End Session (ESID) command and is waiting for a Disconnection indication (N_DISC_IND) from the Network Service.

8.4 Input Events

User Monitor Input Events (Section 3)

F_DATA_RQ	F_CONNECT_RQ	F_START_FILE_RQ	F_CLOSE_FILE_RQ
F_EERP_RQ	F_CONNECT_RS	F_START_FILE_RS(+)	F_CLOSE_FILE_RS(+)
F_CD_RQ	F_ABORT_RQ	F_START_FILE_RS(-)	F_CLOSE_FILE_RS(-)
	F_RELEASE_RQ		

Network Input Events (Section 2.2)

N_CON_IND N_CON_CF N_DATA_IND N_DISC_IND N_RST_IND

Peer ODETTE-FTP Input Events (Section 4)

SSID SFID SFPA SFNA EFID EFPA EFNA
DATA ESID EERP RTR CD CDT SSRM

Internal Input Events

TIME-OUT - Internal ODETTE-FTP timer expires.

Input event parameters are denoted I.Event-name.Parameter-name within the state table action and predicate lists. Their value can be examined but not changed by the ODETTE-FTP entity.

8.5 Output Events

User Monitor Output Events (Section 3)

F_DATA_IND F_CONNECT_IND F_START_FILE_IND F_CLOSE_FILE_IND
F_EERP_IND F_CONNECT_CF F_START_FILE_CF(+) F_CLOSE_FILE_CF(+)
F_CD_IND F_ABORT_IND F_START_FILE_CF(-) F_CLOSE_FILE_CF(-)
F_RELEASE_IND

Network Output Events (Section 2.2)

N_CON_RQ N_CON_RS N_DATA_RQ N_DISC_RQ

Peer ODETTE-FTP Output Events (Section 4)

SSID SFID SFPA SFNA EFID EFPA EFNA
DATA ESID EERP RTR CD CDT SSRM

Output event parameters are denoted O.Event-name.Parameter-name within the state table action and predicate lists. Their values can be examined and changed by the ODETTE-FTP entity.

8.6 Local Variables

The following variables are maintained by the ODETTE-FTP entity to assist the operation of the protocol. They are denoted V.Variable-name within the state table action and predicate lists. Their value can be examined and changed by the ODETTE-FTP entity. The initial value of each variable is undefined.

Variable	Type	Comments
Buf-size	Integer	Negotiated Exchange Buffer size.
Called-addr	Address	Used to build O.F_CONNECT_IND.Called-addr
Calling-addr	Address	To build O.F_CONNECT_IND.Calling-addr
Compression	Yes/No	Compression in used as agreed.
Credit_L	Integer	Listeners credit counter.
Credit_S	Integer	Speaker's credit counter.
Id	String	Used to build O.SSID.Id
Mode		Sender-only, Receiver-only, Both.
Pswd	String	Password, used to build O.SSID.Pswd
Req-buf	Primitive	Input event (F_XXX_RQ) stored in WF_CD state.
Restart	Yes/No	Restart in used as agreed.
Restart-pos	Integer	Used only during file opening.
Window	Integer	The Credit value negotiated for the session.

8.7 Local Constants

The following constants define the capabilities of a given ODETTE-FTP entity. They are denoted C.Constant-name within the state table action and predicate lists. Their value can be examined but not changed by the ODETTE-FTP entity.

Constant	Value	Comments
Cap-compression	Yes/No	Compression supported?
Cap-init	Initiator	Must be Initiator.
	Responder	Must be Responder.
	Both	Can be Initiator or Responder.
Cap-mode	Sender-only	Must be sender.
	Receiver-only	Must be receiver.
	Both	Can be sender or receiver.
Max-buf-size	127 < Int < 100000	Maximum buffer size supported.
Max-window	Int < 1000	Local maximum credit value.

8.8 Session Connection State Table

8.8.1 State Table

Other States									
S	A_WF_CONRS								
T	A_NC_ONLY								
A	I_WF_SSID								
T	I_WF_RM								
E	I_WF_NC								
	IDLE								
	F_CONNECT_RQ	A	X	X	X	X	X	X	X
E	N_CON_CF	X	C	X	X	X	X	X	X
V	SSRM	X	X	H	X	X	X	X	X
E	SSID	X	X	X	D	E	F	F	
N	N_CON_IND	B	X	X	X	X	X	X	X
T	F_CONNECT_RS	X	U	U	U	U	G	U	
	ESID(R=10)	X	X	X	F	X	X	X	X

8.8.2 Transition Table

I	Predicate	Actions	Output Events	Next State
=====O=====				
A	P1: not P1:	1	F_ABORT_IND N_CON_RQ	IDLE I_WF_NC
-----+-----				
B	P3: not P3:		N_DISC_RQ N_CON_RS SSRM	IDLE A_NC_ONLY
-----+-----				
C		2		I_WF_RM
-----+-----				
D	P2: not P2:	4, 2, 5 4, 2	F_CONNECT_CF ESID(R=10) F_ABORT_IND(R,AO=L)	IDLESP WF_NDISC
-----+-----				
E	P4: not P4:	4	N_DISC_RQ F_CONNECT_IND	IDLE A_WF_CONRS
-----+-----				
F			F_ABORT_IND N_DISC_RQ	 IDLE
-----+-----				
G	P2: not P2:	4, 2, 5 4, 2	SSID ESID(R=10) F_ABORT_IND(R,AO=L)	IDLELI WF_NDISC
-----+-----				
H		4, 2, 3	SSID	I_WF_SSID
-----+-----				

8.8.3 Predicates and Actions.

Predicate P1: (No resources available) OR
 (C.Cap-init = Responder) OR
 (C.Cap-mode = Sender-only AND
 I.F_CONNECT_RQ.Mode = Receiver-only) OR
 (C.Cap-mode = Receiver-only AND
 I.F_CONNECT_RQ.Mode = Sender-only)

Predicate P2: Negotiation of (Buf-size, Restart, Compression,
 Mode, Credit) is OK.

Predicate P3: C.Cap-init = Initiator

Predicate P4: Mode in SSID incompatible with C.Cap-mode

- Action 1: Set V.Mode from (C.Cap-mode, I.F_CONNECT_RQ.Mode)
 Set V.Pswd, V.Id, V.Restart from I.F_CONNECT_RQ
 Set V.Buf-size = C.Max-buf-size
 Set V.Compression = C.Cap-compression
 Build O.N_CON_RQ
- Action 2: Start inactivity timer
- Action 3: Set parameters in O.SSID = from local variables
- Action 4: Stop timer
- Action 5: Set V.Mode, V.Restart, V.Compression, V.Buf-size,
 V.Window = from SSID

8.9 Error and Abort State Table

8.9.1 State Table

S T A T E	Other States				
	WF_NDISC				
	I_WF_NC				
	IDLE				
=====O-----+-----+-----+-----					
	TIME-OUT	X	X	A	B
E	F_ABORT_RQ	X	A	X	C
V					
E	N_RST_IND	X	X	A	D
N					
T	N_DISC_IND	X	E	F	G
	Invalid Buffer	X	X	H	I

8.9.2 Transition Table

I	Predicate	Actions	Output Events	Next State
A			N_DISC_RQ	IDLE
B			F_ABORT_IND N_DISC_RQ	IDLE
C		1	N_DISC_RQ	IDLE
D		1	N_DISC_RQ F_ABORT_IND	IDLE
E			F_ABORT_IND	IDLE
F		1		IDLE
G		1	F_ABORT_IND	IDLE
H				WF_NDISC
I		1, 2	ESID(R=01) F_ABORT_IND(R,AO=L)	WF_NDISC

8.9.3 Predicates and Actions.

Action 1: Stop inactivity timer

Action 2: Start inactivity timer

8.10 Speaker State Table 1

8.10.1 State Table

The following abbreviations are used in the Speaker State table.

F_REL_RQ(Ok) - F_RELEASE_RQ Reason = Normal
 F_REL_RQ(Err) - F_RELEASE_RQ Reason = Error

STATE	Other State													
	WF_NDISC													
	OPOWFC													
	OPO													
	OPOP													
	CDSTWFCD													
	SFSTWFCD													
	ERSTWFCD													
	WF_CD													
	WF_RTR													
	IDLESPCD													
	IDLESP													
	EVENT	F_EERP_RQ	A	A	W	F	W	U	U	U	U	U	U	U
F_START_FILE_RQ		B	B	W	G	W	U	U	U	U	U	X	U	
SFPA		C	C	C	C	C	C	C	K	C	C	S	C	
SFNA		C	C	C	C	C	C	C	L	C	C	S	C	
CD		C	C	C	H	R	I	J	C	C	C	S	C	
F_DATA_RQ		U	U	U	U	U	U	U	U	M	V	S	U	
CDT		C	C	C	C	C	C	C	C	P	O	S	C	
F_CD_RQ		D	U	W	T	W	U	U	U	U	U	X	U	
F_REL_RQ(Ok)		U	E	U	U	U	U	U	U	U	U	X	U	
F_REL_RQ(Err)		Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	S	Q	
RTR	C	C	N	C	C	C	C	C	C	C	S	C		

8.10.2 Transition Table

I	Predicate	Actions	Output Events	Next State
A		1,2,3	EERP	WF_RTR
B	P1: not P1:	1,2,5	SFID	UE OPOP
C		1,2	ESID(R=02) F_ABORT_IND(R,AO=L)	WF_NDISC
D		1,2	CD	IDLELICD
E		1,2	ESID(R=00)	WF_NDISC
F		4		ERSTWFCD
G	P1: not P1:	6		UE SFSTWFCD
H		1,2		IDLESP
I		1,2,10	SFID	OPOP
J		1,2	CD	IDLELICD
K	P2: not P2:	1,2 1,2,7,12	ESID(R=02) F_ABORT_IND(R,AO=L) F_START_FILE_CF(+)	WF_NDISC OPO
L		1,2,8	F_START_FILE_CF(-)	IDLESP
M	P3: not P3:	1,2,11,13 1,2,11,13	DATA DATA	OPOWFC OPO
N			Note 3	IDLESP
O		12		OPO See Note 1
P	Protocol Error	1,2	ESID(R=02) F_ABORT_IND(R,AO=L)	WF_NDISC
Q		1,2	ESID(R)	WF_NDISC

Continued -->

I	Predicate	Actions	Output Events	Next State
R		1, 2, 9	EERP	WF_RTR
S				WF_NDISC
T				CDSTWFCD
U			User Error	UE
V			User Error - Note 1	UE
W			User Error - Note 2	UE
X			Error	

8.10.3 Predicates and Actions.

Predicate P1: (I.F_START_FILE_RQ.Restart-pos > 0) AND
((V.Restart = No) OR (V.Mode = Receiver-only))

Note: Restart requested and not supported for this session.

Predicate P2: (I.SFPA.Restart-pos > V.Restart-pos)

Note: Protocol error due to the restart position in the SFPA acknowledgement being greater than the position requested in the SFID request.

Predicate P3: V.Credit_S - 1 = 0

Note: Speaker's Credit is exhausted.

Action 1: Stop inactivity timer

Action 2: Start inactivity timer

Action 3: Build an EERP from F_EERP_RQ

Action 4: Store F_EERP_RQ in V.Req-buf

Action 5: Build SFID from F_START_FILE_RQ
V.Restart-pos = I.F_START_FILE_RQ.Restart-pos

Action 6: Store F_START_FILE_RQ in V.Req-buf

Action 7: Build F_START_FILE_CF(+) from I.SFPA

Action 8: Build F_START_FILE_CF(-) from I.SFNA

Action 9: Build EERP from F_EERP_RQ stored in V.Req-buf

Action 10: Build SFID from F_START_FILE_RQ stored in V.Req-buf
Set V.Restart-pos

Action 11: Build Exchange Buffer

Action 12: V.Credit_S = V.Window

Action 13: V.Credit_S = V.Credit_S - 1

Note 1: The OPOWFC state prevents the Speaker from sending data buffers because it is waiting for credit. The ODETTE-FTP entity may need to control the flow of Data requests (F_DATA_RQ) from it's User Monitor to protect it's own buffers. Any such mechanism and the behaviour of the entity should a User Error occur are regarded as local implementation issues.

Note 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 is considered to be a user error (UE).

Note 3: It is a local matter to make the User Monitor aware that since the RTR is received, the protocol machine is now ready to accept the next request.

8.11 Speaker State Table 2

8.11.1 State Table

S T A T E	CLOP			
	OPOWFC			
	OPO			
	F_CLOSE_FILE_RQ	A	E	U
	EFPA	B	B	C
N T	EFNA	B	B	D

8.11.2 Transition Table

I	Predicate	Actions	Output Events	Next State
A		1,2,5,7	EFID	CLOP
B		1,2	ESID(R=02) F_ABORT_IND(R,AO=L)	WF_NDISC
C	P1:	1,2,3	F_CLOSE_FILE_CF(+,SP=No) CD	IDLELI
	not P1:	1,2,4	F_CLOSE_FILE_CF(+,SP=Yes)	IDLESP
D		1,2,6	F_CLOSE_FILE_CF(-)	IDLESP
E			See Note 1	
U			User Error	UE

8.11.3 Predicates and Actions.

Predicate P1: (I.EFPA.CD-Request = Yes) AND (V.Mode = Both)

Action 1: Stop inactivity timer

Action 2: Start inactivity timer

Action 3: O.F_CLOSE_FILE_CF(+).Speaker = No

Action 4: O.F_CLOSE_FILE_CF(+).Speaker = Yes

Action 5: Build EFID from F_CLOSE_FILE_RQ

Action 6: Build F_CLOSE_FILE_CF(-) from EFNA

Action 7: Set V.Credit_S = 0

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

The ODETTE-FTP implementation must avoid sending EFID (or receiving F_CLOSE_FILE_RQ) while in the OPOWFC state.

8.12 Listener State Table

8.12.1 State Table

S T A T E	CLIP					
	OPI					
	OPIP					
	IDLELICD					
	IDLELI					
E V E N T	SFID	A	A	B	B	B
	DATA	B	B	B	I	B
	EFID	B	B	B	J	B
	F_START_FILE_RS	U	U	H	U	U
	F_CLOSE_FILE_RS	U	U	U	U	K
	CD	C	B	B	B	B
	ESID R=Normal	D	F	D	D	D
	ESID R=Error	D	D	D	D	D
	EERP	E	G	B	B	B

8.12.2 Transition Table

I	Predicate	Actions	Output Events	Next State
=====O=====				
A	P1:	1,2	ESID(R=02) F_ABORT_IND(R,AO=L)	WF_NDISC
	not P1:	1,2,3	F_START_FILE_IND	OPIP
-----+-----				
B		1,2	ESID(R=02) F_ABORT_IND(R,AO=L)	WF_NDISC
-----+-----				
C		1,2	F_CD_IND	IDLESPCD
-----+-----				
D		1	F_ABORT_IND(Received ESID Reason,AO=D) N_DISC_RQ	IDLE
-----+-----				
E		4	F_EERP_IND	
		8	See Note 2 RTR	IDLELI
-----+-----				
F		1	F_RELEASE_IND N_DISC_RQ	IDLE
-----+-----				
G		8	F_EERP_IND See Note 2 RTR	IDLELI
-----+-----				
H	P4:		User Error	UE
	P2,not P4:	1,2	SFPA	OPI
	not(P2,P4):	1,2	SFNA	IDLELI
-----+-----				
I	P5:	1,2	ESID(R=02) F_ABORT_IND(R,A0=L)	WF_NDISC
	not(P5,P6):	1,2,5	F_DATA_IND	OPI
	not P5,P6:	1,2	F_DATA_IND	
		6,7	See Note 1 CDT	OPI
-----+-----				
J		1,2	F_CLOSE_FILE_IND	CLIP
-----+-----				
K	P2,P3:	1,2	EFPA(CD-Req)	WF_CD
	P2,not P3:	1,2	EFPA(no CD)	IDLELI
	not P2:	1,2	EFNA	IDLELI
-----+-----				
U			User Error	UE

8.12.3 Predicates and Actions.

Predicate P1: (I.SFID.Restart-pos > 0) AND (V.Restart = No)

Note: Invalid Start File command

Predicate P2: Positive Response

Predicate P3: I.F_CLOSE_FILE_RS(+).Speaker = Yes

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

Predicate P5: V.Credit_L - 1 < 0

Note: Protocol Error because the Speaker has exceeded it's available transmission credit.

Predicate P6: V.Credit_L - 1 = 0

Note: The Speaker's credit must be reset before it can send further Data Exchange buffers.

Action 1: Stop inactivity timer.

Action 2: Start inactivity timer

Action 3: Build F_START_FILE_IND from I.SFID
V.Restart-pos = I.SFID.Restart-pos

Action 4: Build F_EERP_IND from I.EERP

Action 5: V.Credit_L = V.Credit_L - 1

Action 6: Wait for sufficient resources to receive up to V.Window Data Exchange Buffers.

Action 7: V.Credit_L = V.Window

Action 8: Wait for resources required to process a new EERP.

Note 1: Flow control in case of reception.

The ODETTE-FTP Listener must periodically send new credit to the Speaker. The timing of this operation will depend on:

1. The User Monitor's capacity the receive data.
2. The number of buffers available to ODETTE-FTP.

3. The Speaker's available credit, which must be equal to zero.

Note 2: Generally, the ODETTE-FTP Listener will send RTR immediately after receiving EERP. If required, it can delay the RTR until the resources required to process a new EERP are available.

8.13 Example

Consider an ODETTE-FTP entity that has sent a Start File (SFID) command and entered the Open Out Pending (OPOP) state. Its response on receiving a Positive Answer (SFPA) is documented in Speaker State Table 1 which shows that transition 'K' should be applied and is interpreted as follows:

```

if (I.SFPA.Restart-pos > V.Restart-pos) then
begin
    Actions:      Stop inactivity timer,          // invalid restart
                  Start inactivity timer;         // reset timer
    Output:       ESID(R=02),                      // to peer ODETTE-FTP
                  F_ABORT_IND(R,AO=L);            // to user monitor
    New State:    WF_NDISC;
end
else begin
    Actions:      Stop inactivity timer,          // reset timer
                  Start inactivity timer;
                  Build F_START_FILE_CF(+) from I.SFPA
                  V.Credit_S = V.Window           // initialise credit
    Output:       F_START_FILE_CF(+);              // to user monitor
    New State:    OPO;
end
end

```

The ODETTE-FTP checks the restart position in the received Start File Positive Answer (SFPA) command. If it is invalid it aborts the session by sending an End Session (ESID) command to its peer and an Abort indication (F_ABORT_IND) to its User Monitor. If the restart position is valid a Start File confirmation (F_START_FILE_CF) is built and sent to the User Monitor, the credit window is initialised and the Open Out (OPO) state is entered.

9. Security Considerations

ODETTE-FTP exchanges user identity and password information in clear text. It is therefore recommended that a lower layer (session, network or linkage) security protocol is used to protect the session from casual identity collection.

Appendix A. Virtual File Mapping Example

This example demonstrates the mapping of a Virtual File into a sequence of ODETTE-FTP Data Exchange Buffers and shows how each Stream Transmission Buffer is built from an ODETTE-FTP Data Exchange Buffer prefixed by a Stream Transmission Header.

Each line in this extract from 'The Hunting of the Snark' by Lewis Carroll [SNARK] is considered to be a separate record in a file containing variable length records. Note that it does not represent a text file and CR/LF record separators are not used. The blank line is represented by a zero length record.

```
"It's a Snark!" was the sound that first came to their ears,
    And seemed almost too good to be true.
Then followed a torrent of laughter and cheers:
    Then the ominous words "It's a Boo-"
```

```
Then, silence. Some fancied they heard in the air
    A weary and wandering sigh
Then sounded like "-jum!" but the others declare
    It was only a breeze that went by.
```

Assuming that the minimum exchange buffer length of 128 octets has been negotiated the result of mapping the text into Stream Transmission Buffers may be as follows.

Stream Transmission Buffer 1

```
Text  : ....D."It' s a Snark! " was the sound that first cam
Hex-H  : 10084B2472 7262566762 2276727662 7676627667 2667772666
Hex-L  : 00044C2947 30103E12B1 2071304850 3F5E404814 069234031D
Key    : ----D!.... .....

Text  : e to their ears,. .A nd seemed almost too good to b
Hex-H  : 6276276667 26677242A4 6627666662 6666772766 2666627626
Hex-L  : 504F048592 05123C5061 E40355D540 1CDF3404FF 07FF404F02
Key    : .....!..!.. .....

Text  : e true..Th en followe d a torren t
Hex-H  : 6277762156 6626666676 6262767766 72
Hex-L  : 504255E848 5E06FCCF75 40104F225E 40
Key    : .....!.. .....

Text  : ....D.of l aughter an d cheers:. .Then the ominous w
Hex-H  : 1007496626 6766767266 6266667734 2A56662766 2666667727
Hex-L  : 000847F60C 157845201E 40385523A5 04485E0485 0FD9EF5307
Key    : ----D!.... .....! ..!.....
```

```

Text  : ords "It's a Boo-". Then, silence. Some fancied t
Hex-H : 6767224727 262466228B 5666227666 6662225666 2666666627
Hex-L : F243029473 0102FFD202 485EC039C5 E35E003FD5 061E395404
Key   : .....!! .....

```

```

Text  : hey heard in the air
Hex-H : 6672666762 6627662667
Hex-L : 8590851240 9E04850192
Key   : .....

```

Stream Transmission Buffer 3

```

Text  : ....D. A weary and wandering sigh. Then sounded li
Hex-H : 1007442942 7667726662 7666676662 7666B56662 7676666266
Hex-L : 0008450A10 7512901E40 71E4529E70 39780485E0 3F5E4540C9
Key   : ---D!..!.....!.....

```

```

Text  : ke "-jum!" but the others declare. It was only a
Hex-H : 6622267622 2677276626 7667726666 67642A4727 6726667262
Hex-L : B502DA5D12 025404850F 485230453C 1255029407 130FEC9010
Key   : .....!..!.....

```

```

Text  : breeze that went by.
Hex-H : 6766762766 7276672672
Hex-L : 2255A50481 4075E4029E
Key   : .....

```

Notes:

```

Hex-H      High order bits of octet
Hex-L      Low order bits of octet
Key: ----  Stream Transmission Header
      D     Data Exchange Buffer command code 'D'
      !     Subrecord header octet
      .     Place holder

```

All headers are represented with a period in the Text line.

Each Data Exchange Buffer is preceded by a Stream Transmission Header.

In the above mapping the first Data Exchange Buffer is 128 octets in length. The last record has been continued in the second buffer.

The second Data Exchange Buffer has been truncated at 116 octets to finish at the end of a record. The following record being completely contained in the third buffer. This is an alternative to spanning the record as shown between the first and second Data Exchange Buffers.

The blank line has been encoded as a single header octet of '80' hex, indicating a zero length subrecord with the end of record flag set.

The indented lines have been compressed.

Appendix B. ISO 646 Character Subset

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

Acknowledgements

This document draws extensively on revision 1.3 of the ODETTE File Transfer Specification [OFTP].

Numerous people have contributed to the development of this protocol and their work is hereby acknowledged. The extensions required to utilise the Transmission Control Protocol were formulated and agreed by the current members of ODETTE Working Group Four, who also provided helpful reviews and comments on this document.

References

[OFTP] Organisation for Data Exchange by Tele Transmission in Europe, Odette File Transfer Protocol, Revision 1.3:1993

[RFC-739] Postel, J., Transmission Control Protocol, STD 7, RFC 739, September 1981

[ISO-646] International Organisation for Standardisation, ISO Standard 646:1991, "Information technology -- ISO 7-bit coded character set for information interchange", 1991

[ISO-6523] International Organisation for Standardisation, ISO Standard 6523:1984, "Data interchange -- Structures for the identification of organisations", 1984

[ISO-8601] International Organisation for Standardisation, ISO Standard 8601:1988 "Data elements and interchange formats -- Information interchange -- Representation of dates and times", 1988

[NIFTP] High Level Protocol Group, "A Network Independent File Transfer Protocol", 1981

[SNARK] Carroll, Lewis "The Hunting of the Snark", 1876

ODETTE Address

The ODETTE File Transfer Protocol is a product of Working Group Four of the Organisation for Data Exchange by Tele Transmission in Europe. The working group can be contacted via the ODETTE Secretariat:

ODETTE Secretariat
Forbes House
Halkin Street
London
SW1X 7DS
United Kingdom

Phone: +44 (0)171 344 9227
Fax: +44 (0)171 235 7112
EMail odette@odette.org
keith.oxley@odette.org
stephanie.bioux@odette.org

Author's Address

The author can be contacted at

David Nash
Ford Motor Company Limited
Room 1/148, Central Office
Eagle Way
Warley
Brentwood
Essex
CM13 3BW
United Kingdom

Phone: +44 (0)1277 253043
EMail: dnash@ford.com

