Network Working Group   Ieuan Friend
Internet-Draft   ODETTE
Obsoletes: RFC 2204   April 2007
Category: Informational

ODETTE File Transfer Protocol 2.0
draft-friend-oftp2-04

Status of this Memo

Copyright Notice

Abstract

This memo updates the ODETTE File Transfer Protocol, an established file transfer protocol facilitating electronic data interchange of business data between trading partners, to version 2.0.

The protocol now supports secure and authenticated communication over the Internet using Transport Layer Security, provides file encryption, signing and compression using Cryptographic Message Syntax and provides signed receipts for the acknowledgement of received files.

The protocol supports both direct peer to peer communication and

indirect communication via a Value Added Network and may be used with TCP/IP, X.25 and ISDN based networks.

Table of Contents

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.

The ODETTE-FTP allows business applications to exchange files on a peer to peer basis in a standardised, purely automatic manner and provides a defined acknowledgment process on successful receipt of a file.

ODETTE-FTP is not to be confused as a variant of, or similar to, the Internet FTP [FTP] which provides an interactive means for individuals to share files and which does not have any sort of acknowledgement process. By virtue of its interactive nature, lack of file acknowledgements and client/server design, FTP does not easily lend itself to mission critical environments for the exchange of business data.

Over the last ten years ODETTE-FTP has been widely deployed on systems of all sizes from personal computers to large mainframes while the Internet has emerged as the dominant international network, providing high speed communication at low cost. To match the demand for EDI over the Internet, ODETTE has decided to extend the scope of its file transfer protocol to incorporate security functions and advanced compression techniques to ensure that it remains at the forefront of information exchange technology.

The protocol now supports secure and authenticated communication over the Internet using Transport Layer Security, provides file encryption, signing and compression using Cryptographic Message Syntax and provides signed receipts for the acknowledgement of received files.

The protocol supports both direct peer to peer communication and indirect communication via a Value Added Network and may be used with TCP/IP, X.25 and ISDN based networks.

ODETTE-FTP has been defined by the ODETTE Security Working Group which consists of a number of ODETTE member organisations. All members have significant operational experience working with and developing OFTP and EDI solutions.

1.2 Summary Of Features

This memo is a development of version 1.4 of the ODETTE-FTP [OFTP] with these changes/additions -

- Session level encryption
- File level encryption
- Secure authentication
- File compression
- Signed EERP
- Signed NERP
- Maximum permitted file size increased to 9PB (petabytes)
- Virtual file description added
- Extended error codes

Version 1.4 of ODETTE-FTP included these changes and additions to version 1.3 -

Negative End Response (NERP)
Extended Date and Timestamp
New reason code 14 (File direction refused)

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 of file storage and 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 the 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.

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.

```
                              o---------o
                         Site | Local   |
                          A   | File A  |
                              o---------o
                                   |
      o----------------------- Mapping A ------------------------o
      |                            |                             |
      |                       o---------o                        |
      |                       | Virtual |                        |
      |                       |  File   |                        |
      |                       o---------o                        |
      |    o------------------------------------------------o    |
      |    |                                                |    |
      |    |                  ODETTE-FTP                    |    |
      |    |                                                |    |
      |    o------------------------------------------------o    |
      |      o---------o                        o---------o      |
      |      | Virtual |                        | Virtual |      |
      |      |  File   |                        |  File   |      |
      |      o---------o                        o----+----o      |
      |           |                                  |           |
      o------ Mapping B ------------------------ Mapping C ------o
                  |                                  |
             o---------o                        o----+----o
             | Local   | Site              Site | Local   |
             | File B  |  B                 C   | File C  |
             o---------o                        o---------o
```

A Virtual File is described by a set of attributes identifying and defining the data to be transferred. The main attributes are:

1.5.1 Organisation:

Sequential

Logical records are presented one after another. The ODETTE-FTP must be aware of the record boundaries.

1.5.2 Identification

Dataset Name

Dataset name of the Virtual File being transferred, assigned by bilateral agreement.

Time stamp (HHMMSScccc)

A file qualifier indicating the time the Virtual File was made available for transmission. The counter (cccc=0001-9999) gives higher resolution.

Date stamp (CCYYMMDD)

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

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

The User Monitor may use the Virtual File Date and Time attributes in local processes involving date comparisons and calculations. Any such use falls outside the scope of this protocol.

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 delimit 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 start of 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.

1.7 Security

ODETTE-FTP provides a number of security services to protect a Virtual File transmission across a hostile network.

These security services are as follows:

Confidentiality
Integrity
Non-repudiation of receipt
Non-repudiation of origin
Secure authentication

Security services in this specification are implemented as follows:

Session level encryption
File level encryption
Signed files
Signed receipts
Session level authentication
ODETTE-FTP Authentication

Session level encryption provides data confidentiality by encryption of all the protocol commands and data exchanged between two parties, preventing a third party from extracting any useful information from the transmission.

This session level encryption is achieved by layering ODETTE-FTP over [TLS], Transport Layer Security, distinguishing between secure and unsecure TCP/IP traffic using different port numbers.

File encryption provides complementary data confidentiality by encryption of the files in their entirety. Generally this encryption occurs prior to transmission, but it is also possible to encrypt and send files while in session. File encryption has the additional benefit of allowing a file to remain encrypted outside of the communications session in which it was sent. The file can be received and forwarded by multiple intermediaries, yet only the final destination will be able to decrypt the file. File encryption does not encrypt the actual protocol commands, so trading partner EDI codes and Virtual File Names are still viewable.

Secure authentication is implemented through the session level authentication features available in [TLS] and proves the identity of the parties wishing to communicate.

ODETTE-FTP Authentication also provides an authentication mechanism, but one that is integral to ODETTE-FTP and is available on all network infrastructures over which ODETTE-FTP is operated (this is in contrast to [TLS] which is generally only available over TCP/IP based networks). Both parties are required to possess certificates when ODETTE-FTP Authentication is used.

The security features in ODETTE-FTP 2.0 are centred around the use of [X.509] certificates. To take advantage of the complete range of security services offered in both directions, each party is required to possess an [X.509] certificate. If the confidentiality of data between two parties is the only concern, then [TLS] alone can be used which allows the party accepting an incoming connection (the Responder) to be the only partner required to possess a certificate.

For businesses this means that session level encryption between a hub and its trading partners can be achieved without requiring all the trading partners to obtain a certificate, assuming that trading partners always connect to the hub.

With the exception of [TLS], all the security services work with X.25 and ISDN as transport media. Although nothing technically precludes [TLS] from working with X.25 or ISDN, implementations are rare.

2. Network 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 its OPEN request upon receipt of N_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. Its 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 Secure ODETTE-FTP Session

[TLS] provides a mechanism for securing an ODETTE-FTP session over the Internet or a TCP network. ODETTE-FTP is layered over [TLS], distinguishing between secure and unsecure traffic by using different server ports.

The implementation is very simple. Layer the ODETTE-FTP over [TLS] in the same way as layering ODETTE-FTP over TCP/IP. [TLS] provides both session encryption and authentication, both of which may be used by the connecting parties. A party acts as a [TLS] server when receiving calls and acts as a [TLS] client when making calls. When the [TLS] handshake has completed, the responding ODETTE-FTP may start the ODETTE-FTP session by sending the Ready Message.

2.4 Port Assignment

An 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'.

The responding ODETTE-FTP will listen for secure TLS connections on Registered Port 6619, the service name is 'odette-ftps'.

3. File Transfer Service

The File Transfer Service describes the services offered by an ODETTE-FTP Entity to its User Monitor (generally an application).

NOTE: The implementation of the service primitives is an application
      issue.

3.1 Model

```
       o-------------------o            o-------------------o
       |                   |            |                   |
       |   USER  MONITOR   |            |   USER  MONITOR   |
       |                   |            |                   |
       o-------------------o            o-------------------o
               |   A                            |   A
...............|...|... FILE TRANSFER SERVICE ...|...|...............
               |   |                            |   |
   F_XXX_RQ/RS |   | F_XXX_IND/CF    F_XXX_RQ/RS |   | F_XXX_IND/CF
               V   |                            V   |
       o-------------------o            o-------------------o
       |                   |- - - - - - >|                   |
       | ODETTE-FTP Entity |   E-Buffer  | ODETTE-FTP Entity |
       |                   |< - - - - - -|                   |
       o-------------------o            o-------------------o
               |   A                            |   A
   N_XXX_RQ/RS |   | N_XXX_IND/CF    N_XXX_RQ/RS |   | N_XXX_IND/CF
```

```
                  |   |                             |   |
   ...............|...|...... NETWORK SERVICE ......|...|...............
                  V   |                             V   |
        o---------------------------------------------------------o
        |                                                         |
        |                      N E T W O R K                      |
        |                                                         |
        o---------------------------------------------------------o

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

3.2 Session Setup

3.2.1 Session Connection Service

These diagrams represent the interactions between two communicating ODETTE-FTP entities and their respective User Agents.

The vertical lines represent the ODETTE-FTP entities.
The User Agents are not shown.

```
                             |            |
           F_CONNECT_RQ ---->|------------|----> F_CONNECT_IND
                             |            |
           F_CONNECT_CF <----|------------|<---- F_CONNECT_RS
                             |            |
```

Parameters

```
   Request           Indication        Response          Confirm
   ---------------------------------------------------------------------
   called-address -> same              ---               ----
   calling-address-> same              ---               ----
   ID1 ------------> same              ID2 ------------> same
   PSW1------------> same              PSW2 -----------> same
   mode1 ----------> mode2 ----------> mode3 ----------> same
   restart1 -------> same -----------> restart2 -------> same
   authentication1-> same -----------> authentication2-> 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
---------------------------------------------------------------------
```

Authentication

Specifies the authentication requirement of the User Monitor.

```
   Value:
      Y             Authentication required.
      N             Authentication not required.

   Negotiation:     Not negotiable.

Request            Indication        Response          Confirm
---------------------------------------------------------------------
auth = Y    ----> auth = Y    ----> auth = Y    ----> auth = Y
```

```
   auth = N    ----> auth = N    ----> auth = N    ----> auth = N
   ---------------------------------------------------------------------
```

3.3 File Transfer

3.3.1 File Opening

```
                                |            |
        F_START_FILE_RQ ---->|------------|----> F_START_FILE_IND
                                |            |
   F_START_FILE_CF(+|-) <----|------------|<---- F_START_FILE_RS(+|-)
                                |            |
```

Parameters:

```
   Request          Ind.   RS(+)          CF(+)   RS(-)         CF(-)
   ------------------------------------------------------------------
   filename-------> same   ----           ----    ----          ----
   date-time------> same   ----           ----    ----          ----
   destination----> same   ----           ----    ----          ----
   originator-----> same   ----           ----    ----          ----
   rec-format-----> same   ----           ----    ----          ----
   rec-size ------> same   ----           ----    ----          ----
   file-size------> same   ----           ----    ----          ----
   org-file-size--> same   ----           ----    ----          ----
   signed-eerp----> same   ----           ----    ----          ----
   cipher---------> same   ----           ----    ----          ----
   sec-services---> same   ----           ----    ----          ----
   compression----> same   ----           ----    ----          ----
   envelope-format> same   ----           ----    ----          ----
   description----> 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

```
                                |            |
              F_DATA_RQ ---->|------------|----> F_DATA_IND
                                |            |
              F_DATA_CF <----|(---CDT----)|
```

```
                          |            |
```

Note:

Unlike other commands, where the F_XXX_CF signal is a result of a corresponding F_XXX_RS command, in this case, the local entity layer issues this signal when it is ready for the next data request. This decision is based on the current credit count and the reception of CDT from the receiver.

3.3.3 File Closing

```
                          |            |
      F_CLOSE_FILE_RQ --->|------------|----> F_CLOSE_FILE_IND
                          |            |
 F_CLOSE_FILE_CF(+|-) <---|------------|<---- F_CLOSE_FILE_RS(+|-)
                          |            |
```

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 Listener 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, the Speaker may not send an F_CD_RQ after receiving an unsolicited F_CD_IND. If the Listener receives a solicited F_CD_IND as a result of sending EFPA(Speaker=Yes), it is acceptable to immediately relinquish the right to speak by sending an F_CD_RQ.

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.

```
                          |            |
           F_EERP_RQ ---->|------------|----> F_EERP_IND
                          |            |
           F_RTR_CF  <----|------------|<---- F_RTR_RS
                          |            |
```

Parameters

```
Request               Indication
------------------------------------
filename -----------> same
date ---------------> same
time ---------------> same
destination --------> same
originator ---------> same
hash ---------------> same
signature ----------> 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).

Notes:

1. The F_EERP_RQ (and also F_NERP_RQ) is confirmed with an F_RTR_CF signal. The F_RTR_CF signal is common to both F_EERP_RQ and F_NERP_RQ. There should be no ambiguity, since there can only be one such request pending at any one time.

2. The signature is optional and is requested when sending the F_START_FILE_RQ.

3. If it is not possible to sign the EERP, then an unsigned EERP should still be sent.

4. It is an application implementation issue to validate the contents of the EERP and its signature and to decide what action to take on receipt of an EERP that fails validation or is not signed when a signed EERP was requested.

3.3.6 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 when a file could not be transmitted to the next destination. It is sent only if the problem is of a non-temporary kind.

This service may also be initiated by the final destination instead of sending an End-to-End Response when a file could not be processed, after having successfully received the file.

```
                     |            |
F_NERP_RQ ---->|------------|----> F_NERP_IND
                     |            |
F_RTR_CF  <----|------------|----- F_RTR_RS
                     |            |
```

Parameters

Request   Indication

```
---------------------------------------------------
filename ----------------------> same
date --------------------------> same
time --------------------------> same
destination -------------------> same
originator --------------------> same
creator of negative response --> same
reason ------------------------> same
reason text -------------------> same
hash --------------------------> same
signature ---------------------> same
---------------------------------------------------
```

Relationship with Turn:

The same as for the End-To-End response (see 3.3.5).

3.4 Session Take Down

3.4.1 Normal Close

```
                  |            |
F_RELEASE_RQ ---->|------------|----> F_RELEASE_IND
                  |            |
```

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

```
                  |            |
F_RELEASE_RQ ---->|------------|----> F_ABORT_IND
                  |            |
```

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. Data 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

```
                      |            |
          F_ABORT_RQ ---->|------------|----> F_ABORT_IND
                      |            |
          User Initiated Abort

                      |            |
         F_ABORT_IND <----|------------|----> F_ABORT_IND
                      |            |
         Provider Initiated 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

```
         User  | OFTP |       Network        | OFTP |  User
---------------|------|----------------------|------|---------------
               |      |                      |      |

1. Normal Release

  F_RELEASE_RQ |      | ESID(R=normal)       |      | F_RELEASE_IND
*--------------|->  ==|======================|=>  --|-------------->
  (R=normal)   |      |                      |      |

2. User Initiated Abnormal Release

  F_RELEASE_RQ |      | ESID(R=error)        |      | F_ABORT_IND
*--------------|->  ==|======================|=>   -|-------------->
(R=error value)|      |                      |      | (R=error,AO=D)

         User  | OFTP |       Network        | OFTP |  User
---------------|------|----------------------|------|---------------
               |      |                      |      |

3. Provider Initiated Abnormal Release

  F_ABORT_IND  |      | ESID(R=error)        |      | F_ABORT_IND
<--------------|-*  *=|======================|=>  --|-------------->
               |      |                      |      |

4. User Initiated Connection Abort

 F_ABORT_RQ    |      | N_DISC_RQ            |      | F_ABORT_IND
*--------------|->  --|--------->..----------|->  --|-------------->
               |      |           N_DISC_IND |      | (R=unsp.,AO=D)

5. Provider Initiated Connection Abort

  F_ABORT_IND  |      | N_DISC_RQ            |      | F_ABORT_IND
<--------------|-*  *-|--------->..----------|->  --|-------------->
(R=error,AO=L) |      |           N_DISC_IND |      | (R=unsp.,AO=D)

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

3.5 Service State Automata

These state automata define 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

```
                                 o------------o
                     decision    |            |  f_connect_ind
               +-----------------|    IDLE    |------------------+
               |   F_CONNECT_RQ  |    (0)     |  F_CONNECT_RS    |
               |                 o------------o                  |
               V                                                 |
      o-----------------o                                        |
      |                 |                                        |
      | I_WF_FCONNECTCF |                                        |
      |                 |                                        |
      o--------+--------o                                        |
               |                                                 |
               | F_CONNECT_CF                                    |
               V                                                 V
      o-----------------o                            o-----------------o
      |                 |                            |                 |
      |  IDLE  SPEAKER  |                            | IDLE  LISTENER  |
      |       (1)       |                            |       (2)       |
      |   See Speaker   |                            |  See Listener   |
      |  State Diagram  |                            |  State Diagram  |
      |                 |                            |                 |
      o-----------------o                            o-----------------o
```

3.5.2 Speaker State Diagram

```
   o-----------------o                              o-----------------o
   |  IDLE LISTENER  |                              |      IDLE       |
   | CD_RQ just sent |                              |     see (0)     |
   | see (3), Listen |                              |      Idle       |
   |  State Diagram  |                              |  State Diagram  |
   o-----------------o                              o-----------------o
            A                                                A
            |                                                |
        decision                                          decision
        F_CD_RQ                                         F_RELEASE_RQ
            |                                                |
   o================o decision  o----------o decision  o---------------o
   |                |---------->| WAIT FOR |<----------|               |
   |                | F_EERP_RQ |          | F_EERP_RQ |               |
   |     IDLE       |           | EERP/    |           |    IDLE       |
   |   SPEAKER      | decision  | NERP     | decision  |   SPEAKER     |
   |     (1)        |---------->| CONFIRM. |<----------|     (4)       |
   |                | F_NERP_RQ |          | F_NERP_RQ |               |
   |                |           |          |           |               |
   |                |           |          |           |    CD_IND     |
```

```
   |                | f_rtr_cf  |          |           | just received |
   |                |<----------|          |           |               |
   |                |           o----------o           |               |
   |                |                                  |               |
   |                |                                  |               |
   o================o                                  o---------------o
     A  A        |                                               |
     |  |        | decision and P2              decision and P2  |
     |  |        +-----------------+       +---------------------+
     |  |         F_START_FILE_RQ  |       |    F_START_FILE_RQ
     |  |                          V       V
     |  |                      o---------------o
     |  |  f_file_start_cf(-)  |               |
     |  +----------------------|    OPENING    |
     |                         |               |
     |                         o---------------o
     |                                 |
   f_file_close_cf(-) or          f_start_file_cf(+)
   f_file_close_cf(+) and not P1       |
     |                                 V
   o---------------o     o---------------o  record to send   o---------o
   |               |     |               |------------------>|         |
   |    CLOSING    |     | DATA TRANSFER |     F_DATA_RQ     | NEXT    |
   |               |     |               |                   | RECORD  |
   |               |     |               |     f_data_cf     |         |
   |               |     |               |<------------------|         |
   o---------------o     o---------------o                   o---------o
     |         A                   |
     |         |    end of file    |
     |         +-------------------+
     |            F_CLOSE_FILE_RQ
     |                                              o-----------------o
     |                f_file_close_cf(+) and P1     |  IDLE LISTENER  |
     +---------------------------------------------->| see (2), Listen |
   Predicates:                                      |  State Diagram  |
   P1: Positive confirmation and Speaker = YES      o-----------------o
   P2: Mode = Both or (Mode = Sender-Only)
```

3.5.3 Listener State Diagram

```
   o-----------------o                              o-----------------o
   |  IDLE SPEAKER   |                              |      IDLE       |
   |   CD_IND just   |                              |                 |
   | received see(4) |                              |     see (0)     |
   |  Speaker State  |                              |      Idle       |
   |     Diagram     |                              |  State Diagram  |
   o-----------------o                              o-----------------o
            A                                                A
            |                                                |
         decision      f_eerp_ind                       decision
         F_CD_IND  +---------------+                  F_RELEASE_IND
```

```
         |      |   F_RTR_RS   |                          |
o=================o            |                 o-----------------o
|                 |<-----------+                 |                 |
|                 |                              |                 |
|                 | f_nerp_ind                   |                 |
|                 |------------+                 |                 |
|                 | F_RTR_RS   |                 |                 |
|                 |            |                 |                 |
|                 |<-----------+                 |                 |
|  IDLE LISTENER  |                 f_eerp_ind   |  IDLE LISTENER  |
|       (2)       |<-----------------------------|       (3)       |
|                 |                 F_RTR_RS     |      CD_RQ      |
|                 |                              |    just sent    |
|                 |                 f_nerp_ind   |                 |
|                 |<-----------------------------|                 |
|                 |                 F_RTR_RS     |                 |
|                 |                              |                 |
|                 | f_start_file_ind             |                 |
|                 |    and not P1                |                 |
|                 |---------------------+        |                 |
o=================o F_START_FILE_RS(-)  |        o-----------------o
  A A    |   A  A                       |           |          |
  | |    |   |  +-----------------------+           |          |
  | |    |   |                                      |          |
  | |    |   | f_start_file_ind and not P1          |          |
  | |    |   +--------------------------------------+          |
  | |    |            F_START_FILE_RS(-)                       |
  | |    |                                                     |
  | |    |        f_start_file_ind           f_start_file_ind  |
  | |    |           and P1                        and P1      |
  | |    +----------------------------+     +------------------+
  | |             F_START_FILE_RS(+)  |     | F_START_FILE_RS(+)
  | |                                 V     V
  | |                            o---------------o
  | |f_close_file_ind and not P3 |               |
  | +---------------------------|               |
  |    F_CLOSE_FILE_RS(+,N)      |               |
  |                              |     DATA      |
  |                              |   TRANSFER    |
  |  f_close_file_ind and not P2 |               |-------------+
  +------------------------------|               |             |
       F_CLOSE_FILE_RS(-)        |               |<------------+
                                 o---------------o  F_DATA_IND
o---------------o                           |
| IDLE SPEAKER  |  f_close_file_ind and P3  |
| see (1), Spkr |<--------------------------+
| State Diagram |    F_CLOSE_FILE_RS(+,Y)
o---------------o
```

Predicates:
P1: Decision to send F_START_FILE_RS(+)

P2: Decision to send F_CLOSE_FILE_RS(+)
P3: Decision to become Speaker

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
SECD	Security Change Direction
AUCH	Authentication Challenge
AURP	Authentication Response
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
NERP	Negative 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. Its 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
```

4.2.3 Secure Authentication

Having exchanged SSIDs, the Initiator may optionally begin an authentication phase, in which each party proves its identity to the other.

4.2.4 Protocol Sequence

The first authentication message must be sent by the Initiator.

```
1. Initiator -- SECD ------------> Responder   Change Direction
             <------------ AUCH --             Challenge
             -- AURP ------------>             Response
             <------------ SECD --             Change Direction
             -- AUCH ------------>             Challenge
             <------------ AURP --             Response
```

The Initiator sends a Security Change Direction (SECD) to which the Responder replies with an Authentication Challenge (AUCH).

The Responder looks up the public certificate that is linked to the purported identity of the Initiator (located in the SSID). If the Responder is unable to locate a suitable certificate then authentication fails. The Responder uses the public key contained in the certificate to encrypt a random challenge, unique for each session, for the Initiator. This encrypted challenge is sent as a [CMS] envelope to the Initiator as part of the AUCH.

The Initiator decrypts the challenge using their private key and sends the decrypted challenge back to the Responder in the Authentication Response (AURP).

The Responder checks that the data received in the AURP matches the random challenge that was sent to the Initiator.

If the data matches, then the Initiator has authenticated successfully and the Responder replies with a Security Change Direction (SECD) beginning the complementary process of verifying the Responder to the Initiator. If the data does not match then the Initiator fails authentication.

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
            <------------ NERP --            Negative 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 the Virtual File has been successfully delivered to its final destination. This allows the originator to perform house keeping tasks such as deleting copies of the delivered data.

If the originator of the Virtual File requested a signed EERP in the SFID, the EERP must be signed. Signing allows the originator of the file to prove that the EERP was generated by the final destination. If the final destination is unable to sign the EERP they may send back an unsigned EERP. It is an implementation issue to allow the acceptance of an unsigned EERP if a signed EERP is requested.

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 its 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.

The requesting of a signed EERP is incompatible with the use of broadcast facilities because an EERP can be signed by only one destination. If this scenario occurs, the intermediate broadcast location may continue and ignore the request for a signed EERP or send back a NERP.

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.

```
o----------o                          o-----------o
| Loc. A   |----------- S1 ---------->| Loc. B    |
|          |                          |           |
| [Ba]     |<---------- R2 -----------| [Ba]      |
+----------o                          o-----------o
```

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

Example: Data distribution

Location A sends a Virtual File containing data for distribution to locations B and C via clearing centres E1 and E2. Clearing centre E1 must wait for a response from E2 (for file Ba) and location C before it sends its response, R8, to location A.   Clearing centre E2 can only send response R7 to E1 when location B acknowledges file Ba with response R6.

```
o---------o        o---------o        o---------o        o---------o
| Loc. A  |-- S1 ->| Loc. E1 |-- S2 ->| Loc. E2 |-- S5 ->| Loc. B  |
|         |        |         |        |         |        |         |
| [Ba,Ca] |<- R8 --| [Ba,Ca] |<- R7 --| [Ba]    |<- R6 --| [Ba]    |
o---------o        o---------o        o---------o        o---------o
                      A   |
                      |   |           o---------o
                      |   +----- S3 ->| Loc. C  |
                      |               |         |
                      +--------- R4 --| [Ca]    |
                                      o---------o
```

Example: Data collection

Locations A and B send files Ca and Cb to clearing centre E1 which forwards both files to location C in a single Virtual File. When it receives response R4 from C, clearing centre E1 sends response R5 to location A and R6 to location B.

```
o---------o        o---------o        o---------o
| Loc. A  |-- S1 ->| Loc. E1 |-- S3 ->| Loc. C  |
|         |        |         |        |         |
| [Ca]    |<- R5 --| [Ca,Cb] |<- R4 --| [Ca,Cb] |
o---------o        o---------o        o---------o
                      A   |
o---------o           |   |
| Loc. B  |-- S2 -----+   |
|         |               |
| [Cb]    |<- R6 ---------+
o---------o
```

4.3.7 Negative End Response (NERP)

In addition to the EERP, which allows control over successful transmission of a file, a Negative End Response signals that a file could not be delivered to the final destination or that the final destination could not process the received file.

It may 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 non-temporary, otherwise the intermediate node has to try the transmission again.

It may also be created by the final node that is unable to process the file because of non-recoverable syntax or semantic errors in the file, or because of the failure of any other processing performed on the file.

The NERP will be sent back to the originator of the file.

The parameters are equal to the ones of the EERP, but with additional information about the creator of the NERP and the abort reason. Where the NERP is created due to a failure to transmit, the abort reason is taken from the refusal reason that was sent by the node refusing the file. Because of the NERP it is possible for the intermediate node to stop trying to send the non-deliverable file and to delete the file.

The NERP allows the originator of the file to react to the unsuccessful transmission or processing, depending on the reason code and the creator of the NERP.

If the originator of the Virtual File requested a signed EERP in the SFID, the NERP must be signed. Signing allows the originator of the file to prove by whom the NERP was generated. If the location generating the NERP is unable to sign the NERP they may send back an unsigned NERP. It is an implementation issue to allow the acceptance of an unsigned EERP if a signed NERP is requested.

4.3.8 Ready To Receive Command (RTR)

In order to avoid congestion between two adjacent nodes caused by a continuous flow of EERPs and NERPs, a Ready To Receive (RTR) command is provided. The RTR acts as an EERP/NERP acknowledgement for flow control but has no end-to-end significance.

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

After sending an EERP or NERP, the Speaker must wait for an RTR before sending any other commands. The only acceptable commands to follow are :

EERP
NERP
SFID or CD (if there are no more EERPs or NERPs to be sent)

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
   Listener <------------ NERP -- Speaker    Negative 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. The Speaker may only do this if the Listener has just relinquished its role as speaker.

```
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 by virtue 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], according to Appendix B, is used to encode constants and strings within command exchange buffers except where [UTF-8] is explicitly indicated against a field.

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. Commands can not be compressed. Variable length parameters may be omitted entirely if not required and the associated length indicator field set to zero.

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 fields within a Command Exchange Buffer. Where variable length fields are used, they are preceded with a header field indicating the length. All values are required except where explicitly indicated.

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 SFIDLRECL field may contain any integer value
       between 00000 and 99999.

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

  A String contains alphanumeric characters from the following
  set:

   The numerals:               0 to 9
   The upper case letters:     A to Z
   The following special set:  / - . & ( ) space.

  Space is not allowed as an embedded character.

9(n) - A numeric field of length n octets.

U(n) - A binary field of length n octets.

       Numbers encoded as binary are always unsigned and in
       network byte order.

T(n) - An field of length n octets, encoded using [UTF-8].
```

String and alphanumeric fields are always left justified and right padded with spaces where needed.

Numeric fields are always right justified and left padded with zeros where needed.

Reserved fields should be padded with spaces.

5.3.1 SSRM - Start Session Ready Message

```
o-------------------------------------------------------------------o
|       SSRM        Start Session Ready Message                     |
|                                                                   |
|       Start Session Phase     Initiator <---- Responder            |
|-------------------------------------------------------------------|
| Pos | Field      | Description                           | Format  |
|-----+-----------+---------------------------------------+---------|
|   0 | SSRMCMD    | SSRM Command, 'I'                     | F X(1)  |
```

```
|   1 | SSRMMSG    | Ready Message, 'ODETTE FTP READY '     | F X(17) |
|  18 | SSRMCR     | Carriage Return                        | F X(1)  |
o-------------------------------------------------------------------o
```

SSRMCMD   Command Code   Character

   Value: 'I' SSRM Command identifier.

SSRMMSG   Ready Message   String(17)

   Value: 'ODETTE FTP READY '

SSRMCR   Carriage Return   Character

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

5.3.2 SSID - Start Session

```
o-------------------------------------------------------------------o
|       SSID        Start Session                                   |
|                                                                   |
|       Start Session Phase     Initiator <---> Responder           |
|-------------------------------------------------------------------|
| Pos | Field      | Description                            | Format  |
|-----+------------+----------------------------------------+---------|
|   0 | SSIDCMD    | SSID Command 'X'                       | F X(1)  |
|   1 | SSIDLEV    | Protocol Release Level                 | F 9(1)  |
|   2 | SSIDCODE   | Initiator's Identification Code        | V X(25) |
|  27 | SSIDPSWD   | Initiator's Password                   | V X(8)  |
|  35 | SSIDSDEB   | Data Exchange Buffer Size              | V 9(5)  |
|  40 | SSIDSR     | Send / Receive Capabilities (S/R/B)    | F X(1)  |
|  41 | SSIDCMPR   | Buffer Compression Indicator (Y/N)     | F X(1)  |
|  42 | SSIDREST   | Restart Indicator (Y/N)                | F X(1)  |
|  43 | SSIDSPEC   | Special Logic Indicator (Y/N)          | F X(1)  |
|  44 | SSIDCRED   | Credit                                 | V 9(3)  |
|  47 | SSIDAUTH   | Secure Authentication (Y/N)            | F X(1)  |
|  48 | SSIDRSV1   | Reserved                               | F X(4)  |
|  52 | SSIDUSER   | User Data                              | V X(8)  |
|  60 | SSIDCR     | Carriage Return                        | F X(1)  |
o-------------------------------------------------------------------o
```

SSIDCMD   Command Code   Character

   Value: 'X' SSID Command identifier.

SSIDLEV   Protocol Release Level   Numeric(1)

           Used to specify the level of the ODETTE-FTP protocol

   Value: '1' for Revision 1.2
          '2' for Revision 1.3
          '4' for Revision 1.4

```
          '5' for Revision 2.0

          Future release levels will have higher numbers. The
          protocol release level is negotiable, with the lowest level
          being selected.

          Note: ODETTE File Transfer Protocol 1.3 (RFC2204) specifies
                '1' for the release level, despite adhering to
                revision 1.3.

SSIDCODE  Initiator's Identification Code                  String(25)

 Format:  See Identification Code (Section 5.4)

          Uniquely identifies the Initiator (sender) participating
          in the ODETTE-FTP session.

          It is an application implementation issue to link the
          expected [X.509] certificate to the SSIDCODE provided.

SSIDPSWD  Password                                          String(8)

          Key to authenticate the sender. Assigned by bilateral
          agreement.

SSIDSDEB  Data Exchange Buffer Size                        Numeric(5)

 Minimum: 128
 Maximum: 99999

          The length, in octets, of the largest Data 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 transmissions will not take place in
          the same session.

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

SSIDCMPR  Buffer Compression Indication                     Character
```

Value: 'Y' The location can handle OFTP data buffer compression
'N' The location can not handle OFTP buffer compression

Compression is only used if supported by both locations.

The compression mechanism referred to here applies to each individual OFTP data buffer. This is different from the file compression mechanism in OFTP which involves the compression of whole files.

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: 'Y' Location can handle Special Logic
'N' Location can not handle Special Logic

Special Logic is only used if supported by both locations.

The Special Logic extensions are only useful to access an X.25 network via an asynchronous entry and are not supported for TCP/IP connections.

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".

SSIDAUTH Secure Authentication Character

Value: 'Y' The location requires secure authentication.
'N' The location does not require secure authentication.

Secure authentication is only used if agreed by both locations.

If the answer of the Responder does not match with the authentication requirements of the Initiator, then the Initiator must abort the session.

No negotiation of authentication is allowed.

```
authentication p  -- SSID ------------>
                  <------------ SSID --  authentication q
```

p == q -> continue.
p != q -> abort.

SSIDRSV1 Reserved String(4)

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'.

5.3.3 SFID - Start File

SFID Start File

Start File Phase Speaker ----> Listener

Pos	Field	Description	Format
0	SFIDCMD	SFID Command, 'H'	F X(1)
1	SFIDDSN	Virtual File Dataset Name	V X(26)
27	SFIDRSV1	Reserved	F X(3)
30	SFIDDATE	Virtual File Date stamp, (CCYYMMDD)	V 9(8)
38	SFIDTIME	Virtual File Time stamp, (HHMMSScccc)	V 9(10)
48	SFIDUSER	User Data	V X(8)

56	SFIDDEST	Destination	V X(25)
81	SFIDORIG	Originator	V X(25)
106	SFIDFMT	File Format (F/V/U/T)	F X(1)
107	SFIDLRECL	Maximum Record Size	V 9(5)
112	SFIDFSIZ	File Size, 1K blocks	V 9(13)
125	SFIDOSIZ	Original File Size, 1K blocks	V 9(13)
138	SFIDREST	Restart Position	V 9(17)
155	SFIDSEC	Security Level	F 9(2)
157	SFIDCIPH	Cipher suite selection	F 9(2)
159	SFIDCOMP	File compression algorithm	F 9(1)
160	SFIDENV	File enveloping format	F 9(1)
161	SFIDSIGN	Signed EERP request	F X(1)
162	SFIDDESCL	Virtual File Description length	V 9(3)
165	SFIDDESC	Virtual File Description	V T(n)

SFIDCMD   Command Code   Character

Value: 'H' SFID Command identifier.

SFIDDSN   Virtual File Dataset Name   String(26)

Dataset name of the Virtual File being transferred, assigned by bilateral agreement.

No general structure is defined for this attribute.

See Virtual Files - Identification (Section 1.5.2)

SFIDRSV1   Reserved   String(3)

This field is reserved for future use.

SFIDDATE   Virtual File Date stamp   Numeric(8)

Format: 'CCYYMMDD' 8 decimal digits representing the century, year, month and day.

Date 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)

SFIDTIME   Virtual File Time stamp   Numeric(10)

Format: 'HHMMSScccc' 10 decimal digits representing hours, minutes, seconds and a counter (0001-9999), which gives higher resolution

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)

Once a file has been signed, compressed and/or encrypted, in file format terms it becomes unstructured, format U. The record boundaries are no longer discernable until the file is decrypted, decompressed and/or verified. SFID File Format Field in this scenario indicates the format of the original file and the transmitted file must be treated as U format.

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'.

If SFIDFMT is 'V' and the file is compressed, encrypted or signed then the maximum value of SFIDRECL is '65536'.

SFIDFSIZ Transmitted File Size Numeric(13)

Maximum: 9999999999999

Space in 1K (1024 octet) blocks required at the Originator location to store the actual Virtual File that is to be transmitted.

e.g. if a file is compressed before sending, then this is the space required to store the compressed file.

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

13 digits allows for a maximum file size of approximately 9.3PB (petabytes) to be transmitted.

SFIDOSIZ Original File Size Numeric(13)

Maximum: 9999999999999

Space in 1K (1024 octet) blocks required at the Originator location to store the original before it was signed, compressed and/or encrypted.

If no security or compression services have been used, SFIDOSIZ should contain the same value as SFIDFSIZ.

If the original file size is not known, the value zero should be used.

This parameter is intended to provide only a good estimate of the original file size.

The sequence of events in file exchange are:

(a) raw data file ready to be sent
SFIDOSIZ = Original File Size

(b) signing/compression/encryption

(c) transmission
SFIDFSIZ = Transmitted File Size

(d) decryption/decompression/verification

(e) received raw data file for in-house applications
SFIDOSIZ = Original File Size

The Transmitted File Size at (c) indicates to the receiver how much storage space is needed to receive the file.

The Original File Size at (e) indicates to the in-house application how much storage space is needed to process the file.

SFIDREST   Restart Position   Numeric(17)

Maximum: 99999999999999999

Virtual File restart position.

The count represents the:
- Record Number if SSIDFMT is 'F' or 'V'.
- File offset in 1K (1024 octet) blocks if SFIDFMT is 'U' or 'T'.

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

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

Once a file has been signed, compressed and/or encrypted, in file format terms, it has become unstructured, like format U. The file should be treated as format U for the purposes of restart, regardless of the actual value in SFIDFMT.

SFIDSEC   Security Level   Numeric(2)

Value: '00' No security services
'01' Encrypted
'02' Signed
'03' Encrypted and signed

Indicates whether the file has been signed and/or encrypted before transmission. (See Section 6.2)

SFIDCIPH   Cipher suite selection   Numeric(2)

Value: '00' No security services
'01' See Section 10.2

Indicates the cipher suite used to sign and/or encrypt the file and also to indicate the cipher suite that should

```
           be used when a signed EERP or NERP is requested.

 SFIDCOMP  File compression algorithm                    Numeric(1)

    Value: '0'  No compression
           '1'  Compressed with [ZLIB] algorithm

           Indicates the algorithm used to compress the file.
           (See Section 6.4)

 SFIDENV   File enveloping format                        Numeric(1)

    Value: '0'  No envelope
           '1'  File is enveloped using [CMS]

           Indicates the enveloping format used in the file.

           If the file is encrypted/signed/compressed or is an
           enveloped file for the exchange and revocation of
           certificates, this field must be set accordingly.

 SFIDSIGN  Signed EERP request                            Character

    Value: 'Y'  The EERP returned in acknowledgement of the file
                must be signed
           'N'  The EERP must not be signed

           Requests whether the EERP returned for the file must
           be signed.

 SFIDDESCL Virtual File Description length               Numeric(3)

           Length in octets of the field SFIDDESC.

           A value of 0 indicates that no description is present.

 SFIDDESC  Virtual File Description                      [UTF-8](n)

           May be used by the ODETTE-FTP in any way. If not used,
           SFIDDESCL should be set to zero.

           No general structure is defined for this attribute but it
           is expected that a bilateral agreement exists as to the
           meaning of the data.

           It is encoded using [UTF-8] to support a range of national
           languages.

           Maximum length of the encoded value is 999 octets.
```

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(17)

SFPACMD   Command Code   Character

Value: '2' SFPA Command identifier.

SFPAACNT   Answer Count   Numeric(17)

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)
4	SFNAREASL	Answer Reason Text Length	V 9(3)
7	SFNAREAST	Answer Reason Text	V T(n)

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.
          '14'  File direction refused.
          '15'  Cipher suite not supported.
          '16'  Encrypted file not allowed.
          '17'  Unencrypted file not allowed.
          '18'  Compression not allowed.
          '19'  Signed file not allowed.
          '20'  Unsigned file not allowed.
          '99'  Unspecified reason.

          Reason why transmission can not proceed.

SFNARRTR  Retry Indicator                                    Character

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

          This parameter is used to advise the Speaker if it should
          retry at a later 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.

SFNAREASL Answer Reason Text Length                        Numeric(3)

          Length in octets of the field SFNAREAST.

          0 indicates that no SFNAREAST field follows.

SFNAREAST Answer Reason Text                               [UTF-8](n)

          Reason why transmission can not proceed in plain text.

          It is encoded using [UTF-8].

          Maximum length of the encoded reason is 999 octets.

          No general structure is defined for this attribute.
```

5.3.6 DATA - Data Exchange Buffer

```
o-------------------------------------------------------------------o
|       DATA        Data Exchange Buffer                            |
```

```
|                                                                   |
|       Data Transfer Phase        Speaker ----> Listener            |
|-------------------------------------------------------------------|
| Pos | Field      | Description                           | Format   |
|-----+-----------+----------------------------------------+---------|
|   0 | DATACMD    | DATA Command, 'D'                     | F X(1)   |
|   1 | DATABUF    | Data Exchange Buffer payload          | V U(n)   |
o-------------------------------------------------------------------o
```

```
DATACMD   Command Code                                     Character

   Value: 'D'  DATA Command identifier.

DATABUF   Data Exchange Buffer payload                     Binary(n)

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

5.3.7 CDT - Set Credit

```
o-------------------------------------------------------------------o
|       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)   |
o-------------------------------------------------------------------o
```

```
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

```
o-------------------------------------------------------------------o
|       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(17)  |
|  18 | EFIDUCNT   | Unit Count                            | V 9(17)  |
o-------------------------------------------------------------------o
```

```
   EFIDCMD   Command Code                                      Character

      Value: 'T'  EFID Command identifier.

   EFIDRCNT  Record Count                                    Numeric(17)

    Maximum: 99999999999999999

             For SSIDFMT 'F' or 'V' the exact record count.
             For SSIDFMT 'U' or 'T' zeros.

             The count will express the real size of the file (before
             buffer compression, header not included). The total count
             is always used, even during restart processing.

   EFIDUCNT  Unit Count                                      Numeric(17)

    Maximum: 99999999999999999

             Exact number of units (octets) transmitted.

             The count will express the real size of the file. The
             total count is always used, even during restart processing.
```

5.3.9 EFPA - End File Positive Answer

```
   o-------------------------------------------------------------------o
   |       EFPA        End File Positive Answer                        |
   |                                                                   |
   |       End File Phase             Speaker <---- Listener           |
   |-------------------------------------------------------------------|
   | Pos | Field      | Description                           | Format   |
   |-----+-----------+---------------------------------------+---------|
   |   0 | EFPACMD    | EFPA Command, '4'                     | F X(1)  |
   |   1 | EFPACD     | Change Direction Indicator, (Y/N)     | F X(1)  |
   o-------------------------------------------------------------------o

   EFPACMD   Command Code                                      Character

      Value: '4'  EFPA Command identifier.

   EFPACD    Change Direction Indicator                        Character

      Value: 'N'  Change direction not requested.
             'Y'  Change direction requested.

             This parameter allows the Listener to request a Change
             Direction (CD) command from the Speaker.
```

5.3.10 EFNA - End File Negative Answer

```
o-------------------------------------------------------------------o
|       EFNA        End File Negative Answer                        |
|                                                                   |
|       End File Phase             Speaker <---- Listener           |
|-------------------------------------------------------------------|
| Pos | Field      | Description                          | Format   |
|-----+------------+--------------------------------------+----------|
|   0 | EFNACMD    | EFNA Command, '5'                    | F X(1)   |
|   1 | EFNAREAS   | Answer Reason                        | F 9(2)   |
|   3 | EFNAREASL  | Answer Reason Text Length            | V 9(3)   |
|   6 | EFNAREAST  | Answer Reason Text                   | V T(n)   |
o-------------------------------------------------------------------o
```

```
EFNACMD   Command Code                                     Character

   Value: '5'  EFNA Command identifier.

EFNAREAS  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.
          '14'  File direction refused.
          '15'  Cipher suite not supported.
          '16'  Encrypted file not allowed.
          '17'  Unencrypted file not allowed.
          '18'  Compression not allowed.
          '19'  Signed file not allowed.
          '20'  Unsigned file not allowed.
          '21'  Invalid file signature.
          '22'  File decryption failure.
          '23'  File decompression failure.
          '99'  Unspecified reason.

          Reason why transmission failed.

EFNAREASL Answer Reason Text Length                       Numeric(3)

          Length in octets of the field EFNAREAST.

          0 indicates that no EFNAREAST field follows.

EFNAREAST Answer Reason Text                              [UTF-8](n)

          Reason why transmission failed in plain text.
```

It is encoded using [UTF-8].

Maximum length of the encoded reason is 999 octets.

No general structure is defined for this attribute.

5.3.11 ESID - End Session

```
o-------------------------------------------------------------------o
|       ESID        End Session                                     |
|                                                                   |
|       End Session Phase          Speaker ----> Listener            |
|-------------------------------------------------------------------|
| Pos | Field     | Description                           | Format  |
|-----+-----------+---------------------------------------+---------|
|   0 | ESIDCMD   | ESID Command, 'F'                     | F X(1)  |
|   1 | ESIDREAS  | Reason Code                           | F 9(2)  |
|   3 | ESIDREASL | Reason Text Length                    | V 9(3)  |
|   6 | ESIDREAST | Reason Text                           | V T(n)  |
|     | ESIDCR    | Carriage Return                       | F X(1)  |
o-------------------------------------------------------------------o
```

ESIDCMD   Command Code   Character

   Value: 'F' ESID Command identifier.

ESIDREAS   Reason Code   Numeric(2)

   Value   '00' Normal session termination

           '01' Command not recognised

                 An Exchange Buffer contains an invalid command code (1st octet of the buffer).

           '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 differs from 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

'11' Invalid challenge response

'12' Secure authentication requirements incompatible

'99' Unspecified Abort code

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

ESIDREASL Reason Text Length Numeric(3)

Length in octets of the field ESIDREAST.

0 indicates that no ESIDREAST field is present.

ESIDREAST Reason Text [UTF-8](n)

Reason why session ended in plain text.

It is encoded using [UTF-8].

Maximum length of the encoded reason is 999 octets.

No general structure is defined for this attribute.

ESIDCR Carriage Return Character

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

5.3.12 CD - Change Direction

```
o-------------------------------------------------------------------o
|       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)  |
o-------------------------------------------------------------------o
```

```
CDCMD     Command Code                                      Character

   Value: 'R'  CD Command identifier.
```

5.3.13 EERP - End to End Response

```
o-------------------------------------------------------------------o
|       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(3)  |
|  30 | EERPDATE  | Virtual File Date stamp, (CCYYMMDD)   | V 9(8)  |
|  38 | EERPTIME  | Virtual File Time stamp, (HHMMSScccc) | V 9(10) |
|  48 | EERPUSER  | User Data                             | V X(8)  |
|  56 | EERPDEST  | Destination                           | V X(25) |
|  81 | EERPORIG  | Originator                            | V X(25) |
| 106 | EERPHSHL  | Virtual File Hash length              | V U(2)  |
| 108 | EERPHSH   | Virtual File Hash                     | V U(n)  |
|     | EERPSIGL  | EERP signature length                 | V U(2)  |
|     | EERPSIG   | EERP signature                        | V U(n)  |
o-------------------------------------------------------------------o
```

```
EERPCMD   Command Code                                      Character

   Value: 'E'  EERP Command identifier.

EERPDSN   Virtual File Dataset Name                        String(26)

          Dataset name of the Virtual File being transferred,
          assigned by bilateral agreement.
```

No general structure is defined for this attribute.

See Virtual Files - Identification (Section 1.5.2)

EERPRSV1 Reserved String(3)

This field is reserved for future use.

EERPDATE Virtual File Date stamp Numeric(8)

Format: 'CCYYMMDD' 8 decimal digits representing the century, year, month and day respectively.

Date 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)

EERPTIME Virtual File Time stamp Numeric(10)

Format: 'HHMMSScccc' 10 decimal digits representing hours, minutes, seconds and a counter (0001-9999), which gives higher resolution

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)

EERPUSER 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.

EERPDEST Destination String(25)

Format: See Identification Code (Section 5.4)

Originator of the Virtual File.

This is the location that created the data for transmission.

EERPORIG Originator String(25)

Format: See Identification Code (Section 5.4)

Final Recipient of the Virtual File.

This is the location that will look into the Virtual File content and process it accordingly. It is also the location that creates the EERP for the received file.

EERPHSHL Virtual File hash length Binary(2)

Length in octets of the field EERPHSH.

A binary value of 0 indicates that no hash is present. This is always the case if the EERP is not signed.

EERPHSH Virtual File hash Binary(n)

Hash of the transmitted Virtual File. i.e. not the hash of the original file.

The algorithm used is determined by the bilaterally agreed cipher suite specified in the SFIDCIPH.

It is an application implementation issue to validate the EERPHSH to ensure that the EERP is acknowledging the exact same file as was originally transmitted.

EERPSIGL EERP Signature length Binary(2)

0 indicates that this EERP has not been signed.

Any other value indicates the length of EERPSIG in octets and indicates that this EERP has been signed.

EERPSIG EERP Signature Binary(n)

Contains the [CMS] enveloped signature of the EERP.

```
Signature = Sign{EERPDSN
                 EERPDATE
                 EERPTIME
                 EERPDEST
                 EERPORIG
                 EERPHSH}
```

Each field is taken in its entirety, including any padding. The envelope must contain the original data, not just the signature.

The [CMS] content type used is SignedData.

The encapsulated content type used is id-data.

It is an application issue to validate the signature with the contents of the EERP.

5.3.14 NERP - Negative End Response

```
o-------------------------------------------------------------------o
|       NERP        Negative End Response                           |
|                                                                   |
|       Start File Phase           Speaker ----> Listener            |
|       End File Phase             Speaker ----> Listener            |
|-------------------------------------------------------------------|
| Pos | Field     | Description                           | Format  |
|-----+-----------+---------------------------------------+---------|
|   0 | NERPCMD   | NERP Command, 'N'                     | F X(1)  |
|   1 | NERPDSN   | Virtual File Dataset Name             | V X(26) |
|  27 | NERPRSV1  | Reserved                              | F X(6)  |
|  33 | NERPDATE  | Virtual File Date stamp, (CCYYMMDD)   | V 9(8)  |
|  41 | NERPTIME  | Virtual File Time stamp, (HHMMSScccc) | V 9(10) |
|  51 | NERPDEST  | Destination                           | V X(25) |
|  76 | NERPORIG  | Originator                            | V X(25) |
| 101 | NERPCREA  | Creator of NERP                       | V X(25) |
| 126 | NERPREAS  | Reason code                           | F 9(2)  |
| 128 | NERPREASL | Reason text length                    | V 9(3)  |
| 131 | NERPREAST | Reason text                           | V T(n)  |
|     | NERPHSHL  | Virtual File hash length              | V U(2)  |
|     | NERPHSH   | Virtual File hash                     | V U(n)  |
|     | NERPSIGL  | NERP signature length                 | V U(2)  |
|     | NERPSIG   | NERP signature                        | V U(n)  |
o-------------------------------------------------------------------o
```

NERPCMD   Command Code   Character

   Value: 'N' NERP Command identifier.

NERPDSN   Virtual File Dataset Name   String(26)

Dataset name of the Virtual File being transferred, assigned by bilateral agreement.

No general structure is defined for this attribute.

See Virtual Files - Identification (Section 1.5.2)

NERPRSV1 Reserved   String(6)

This field is reserved for future use.

NERPDATE Virtual File Date stamp   Numeric(8)

  Format: 'CCYYMMDD' 8 decimal digits representing the century, year, month and day respectively.

Date 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)

```
NERPTIME  Virtual File Time stamp                        Numeric(10)

   Format: 'HHMMSScccc'  10 decimal digits representing hours,
          minutes, seconds and a counter (0001-9999), which gives
          higher resolution

          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)

NERPDEST  Destination                                     String(25)

  Format: See Identification Code (Section 5.4)

          Originator of the Virtual File.

          This is the location that created the data for
          transmission.

NERPORIG  Originator                                      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.

NERPCREA  Creator of the NERP                             String(25)

  Format: See Identification Code (Section 5.4)

          It is the location that created the NERP.

NERPREAS  Reason code                                     Numeric(2)

          This attribute will specify why transmission cannot
          proceed or why processing of the file failed.

          "SFNA(RETRY=N)" below should be interpreted as "EFNA or
          SFNA(RETRY=N)" where appropriate.

   Value  '03'  ESID received with reason code '03'
                 ( user code not known )
          '04'  ESID received with reason code '04'
                 ( invalid password )
          '09'  ESID received with reason code '99'
```

```
       ( unspecified reason )
'11'  SFNA(RETRY=N) received with reason code '01'
       ( invalid file name )
'12'  SFNA(RETRY=N) received with reason code '02'
       ( invalid destination )
'13'  SFNA(RETRY=N) received with reason code '03'
       ( invalid origin )
'14'  SFNA(RETRY=N) received with reason code '04'
       ( invalid storage record format )
'15'  SFNA(RETRY=N) received with reason code '05'
       ( maximum record length not supported )
'16'  SFNA(RETRY=N) received with reason code '06'
       ( file size too big )
'20'  SFNA(RETRY=N) received with reason code '10'
       ( invalid record count )
'21'  SFNA(RETRY=N) received with reason code '11'
       ( invalid byte count )
'22'  SFNA(RETRY=N) received with reason code '12'
       ( access method failure )
'23'  SFNA(RETRY=N) received with reason code '13'
       ( duplicate file )
'24'  SFNA(RETRY=N) received with reason code '14'
       ( file direction refused )
'25'  SFNA(RETRY=N) received with reason code '15'
       ( cipher suite not supported )
'26'  SFNA(RETRY=N) received with reason code '16'
       ( encrypted file not allowed )
'27'  SFNA(RETRY=N) received with reason code '17'
       ( unencrypted file not allowed )
'28'  SFNA(RETRY=N) received with reason code '18'
       ( compression not allowed)
'29'  SFNA(RETRY=N) received with reason code '19'
       ( signed file not allowed)
'30'  SFNA(RETRY=N) received with reason code '20'
       (unsigned file not allowed)
'31'  File signature not valid.
'32'  File decompression failed.
'33'  File decryption failed.
'34'  File processing failed.
'35'  Not delivered to recipient.
'36'  Not acknowledged by recipient.
'50'  Transmission stopped by the operator.
'90'  File size incompatible with recipient's
       protocol version
'99'  Unspecified reason.
```

NERPREASL Reason Text Length   Numeric(3)

Length in octets of the field NERPREAST.

0 indicates that no NERPREAST field follows.

```
NERPREAST Reason Text                                  [UTF-8](n)

          Reason why transmission cannot proceed in plain text.

          It is encoded using [UTF-8].

          Maximum length of the encoded reason is 999 octets.

          No general structure is defined for this attribute.

NERPHSHL  Virtual File hash length                       Binary(2)

          Length in octets of the field NERPHSH.

          A binary value of 0 indicates that no hash is present.
          This is always the case if the NERP is not signed.

NERPHSH   Virtual File hash                              Binary(n)

          Hash of the Virtual File being transmitted.

          The algorithm used is determined by the bilaterally agreed
          cipher suite specified in the SFIDCIPH.

NERPSIGL  NERP Signature length                          Binary(2)

          0 indicates that this NERP has not been signed.

          Any other value indicates the length of NERPSIG in octets
          and indicates that this NERP has been signed.

NERPSIG   NERP Signature                                 Binary(n)

          Contains the [CMS] enveloped signature of the NERP.

          Signature = Sign{NERPDSN
                           NERPDATE
                           NERPTIME
                           NERPDEST
                           NERPORIG
                           NERPCREA
                           NERPHSH}

          Each field is taken in its entirety, including any
          padding. The envelope must contain the original data,
          not just the signature.

          The [CMS] content type used is SignedData.

          The encaulated content type used is id-data.

          It is an application issue to validate the signature with
```

          the contents of the NERP.

5.3.15 RTR - Ready To Receive

```
   o-------------------------------------------------------------------o
   |       RTR         Ready To Receive                                |
   |                                                                   |
   |       Start File Phase         Initiator <---- Responder          |
   |       End File Phase           Initiator <---- Responder          |
   |-------------------------------------------------------------------|
   | Pos | Field      | Description                           | Format |
   |-----+------------+---------------------------------------+--------|
   |   0 | RTRCMD     | RTR Command, 'P'                      | F X(1) |
   o-------------------------------------------------------------------o
```

   RTRCMD   Command Code   Character

      Value: 'P' RTR Command identifier.

5.3.16 SECD - Security Change Direction

```
   o-------------------------------------------------------------------o
   |       SECD        Security Change Direction                       |
   |                                                                   |
   |       Start Session Phase     Initiator <---> Responder           |
   |-------------------------------------------------------------------|
   | Pos | Field      | Description                           | Format |
   |-----+------------+---------------------------------------+--------|
   |   0 | SECDCMD    | SECD Command, 'J'                     | F X(1) |
   o-------------------------------------------------------------------o
```

   SECDCMD   Command Code   Character

      Value: 'J' SECD Command identifier.

5.3.17 AUCH - Authentication Challenge

```
   o-------------------------------------------------------------------o
   |       AUCH        Authentication Challenge                        |
   |                                                                   |
   |       Start Session Phase     Initiator <---> Responder           |
   |-------------------------------------------------------------------|
   | Pos | Field      | Description                           | Format |
   |-----+------------+---------------------------------------+--------|
   |   0 | AUCHCMD    | AUCH Command, 'A'                     | F X(1) |
   |   1 | AUCHCHLL   | Challenge Length                      | V U(2) |
   |   3 | AUCHCHAL   | Challenge                             | V U(n) |
   o-------------------------------------------------------------------o
```

   AUCHCMD   Command Code   Character

      Value: 'A' AUCH Command identifier.

```
AUCHCHLL  Challenge length                                 Binary(2)

          Indicates the length of AUCHCHAL in octets.

          The length is expressed as an unsigned binary number using
          network byte order.

AUCHCHAL  Challenge                                        Binary(n)

          A [CMS] encrypted 20 byte random number uniquely generated
          each time an AUCH is sent.
```

5.3.18 AURP - Authentication Response

```
o-------------------------------------------------------------------o
|       AURP        Authentication Response                         |
|                                                                   |
|       Start Session Phase     Initiator <---> Responder            |
|-------------------------------------------------------------------|
| Pos | Field      | Description                         | Format   |
|-----+------------+-------------------------------------+----------|
|   0 | AURPCMD    | AURP Command, 'S'                   | F X(1)   |
|   1 | AURPRSP    | Response                            | V U(20)  |
o-------------------------------------------------------------------o
```

```
AURPCMD   Command Code                                     Character

   Value: 'S'  AURP Command identifier.

AURPRSP   Response                                        Binary(20)

          Contains the decrypted challenge (AUCHCHAL).

          IMPORTANT:

          It is an application implementation issue to validate a
          received AURP to ensure that the response matches the
          challenge. This validation is extremely important to ensure
          that a party is correctly authenticated.
```

5.4 Identification Code

The Initiator (sender) and Responder (receiver) participating in an ODETTE-FTP session are uniquely identified by an Identification Code based on [ISO-6523], Structure for the Identification of Organisations (SIO). The locations are considered to be adjacent for the duration of the transmission.

The SIO has the following format.

```
o-------------------------------------------------------------------o
```

Pos	Field	Description	Format
0	SIOOID	ODETTE Identifier	F X(1)
1	SIOICD	International Code Designator	V 9(4)
5	SIOORG	Organisation Code	V X(14)
19	SIOCSA	Computer Sub-Address	V X(6)

SIOOID ODETTE Identifier Character

Value: 'O' Indicates ODETTE assigned Organisation Identifier.
Other values may be used for non-ODETTE codes.

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, space 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. File Services

6.1 Overview

The ODETTE-FTP provides services for compressing, encrypting and signing files. These services should generally be performed off line, outside of the ODETTE-FTP communications session for performance reasons although this is not a strict requirement.

The ODETTE-FTP requires that the following steps must be performed in this exact sequence, although any of steps 2, 3 or 4 may be omitted. Step 1 is required only if any of steps 2, 3, or 4 are performed:

1. Insert record length indicators (V Format files only)(Section 6.5)
2. Sign
3. Compress
4. Encrypt

The cipher suite for the encryption and signing algorithms is assigned by bilateral agreement.

Secured and/or compressed files must be enveloped. The envelope contains additional information about the service used that is necessary for a receiving party to fully process the file.

The [CMS] content types used are:

```
EnvelopedData  - Indicates encrypted data
CompressedData - Indicates compressed data
SignedData     - Indicates signed content
Data           - Indicates unstructured data
```

For signed or encrypted data, the encapsulated content type (eContentType field) is id-data.

6.2 File Signing

Files that are to be signed are enveloped according to the file enveloping format (SFIDENV). Generally this will be as a [CMS] package.

A file may be signed more than once to ease the changeover between old and new certificates.

It is recommended that the envelope does not contain the public certificate of the signer. Where files are sent to the same recipient continuously, it would serve no benefit to repeatedly send the same certificate. Both the orginal file data and signature are stored within the [CMS] package.

6.3 File Encryption

Files that are to be encrypted are enveloped according to the file enveloping format (SFIDENV). Generally this will be as a [CMS] package.

It is recommended that encryption should be performed before the ODETTE-FTP session starts because a large file takes a long time to encrypt and could cause session time outs, even on high performance machines.

Likewise, decryption of the file should occur outside of the session. Though it may be that an application chooses to allow in-session encryption and decryption for very small files.

6.4 File Compression

Files that are to be compressed are enveloped according to the file enveloping format (SFIDENV). Generally this will be as a [CMS] package using the [CMS Compression] data type, which uses the [ZLIB] compression algorithm by default.

Unlike the buffer compression method, this method operates on a whole file. Because of the increased levels of compression, file level compression essentially deprecates the older buffer compression inside ODETTE-FTP. The buffer compression is kept for backwards compatibility.

6.5 V Format Files - Record Lengths

A file that has been signed, compressed and/or encrypted will have lost its record structure, so ODETTE-FTP will not be able to insert the End of Record Flag in sub record headers in Data Exchange Buffers. To preserve the record structure, V format files must have record headers inserted into them prior to signing, compression or encryption. These 2 byte binary numbers, in network byte order, indicate the length of each record, allowing the receiving system, where appropriate, to recreate the files complete with the original variable length records. Note that the header bytes hold the number of data bytes in the record and don't include themselves.

This is only applicable to V Format files, which themselves are typically only of concern for mainframes.

7. ODETTE-FTP Data Exchange Buffer

7.1 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.

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.

7.2 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,

```
o-------------------------------------------------------
| C | H |           | H |           | H |           |   /
| M | D | SUBRECORD | D | SUBRECORD | D | SUBRECORD |  /_
| D | R |           | R |           | R |           |   /
o------------------------------------------------------
```

CMD

The Data Exchange Buffer Command Character, 'D'.

HDR

A one octet Subrecord Header defined as follows:

```
  0   1   2   3   4   5   6   7
o-------------------------------o
| E | C |                       |
| o | F | C O U N T             |
| R |   |                       |
o-------------------------------o
```

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.

7.3 Buffer Filling Rules

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

Virtual File records may be concatenated within one Data 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.

8. Stream Transmission Buffer

8.1 Introduction

To utilise the TCP stream 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.

Note: The Stream Transmission Buffer is not used when using ODETTE-FTP over an X.25 network.

This is because ODETTE-FTP can rely 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. TCP offers a stream based connection which does not provide these functions.

The Stream Transmission Buffer comprises of a STH and OEB.

```
o-----+-----------------+-----+-------------------+-----+------
| STH | OEB             | STH |  OEB              | STH | OEB/
o-----+-----------------+-----+-------------------+-----+----
```

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

8.2 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 in network byte order.

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.

9. Protocol State Machine

9.1 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.

9.2 Error Handling

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

1. The case is impossible by design 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.

9.3 States

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

A_NC_ONLY   Responder, Network Connection opened

The Responder has sent its 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 its 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 its 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.

NRSTWFCD   Negative End Response stored in WF_CD state

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

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 its 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 its 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.

RTRP   Ready to Receive (RTR) Pending

The Listener has received an EERP or a NERP and is waiting for the Ready to Receive response (F_RTR_RS) from its User Monitor.

SFSTWFCD   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 Speaker has sent an End to End Response (EERP) or a Negative End Response (NERP) command and must wait for Ready To Receive (RTR) from the Listener.

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.

WF_SECD   Wait for Security Change Direction

The Speaker is expecting a Security Change Direction (SECD) from the Listener.

WF_AUCH   Wait for Authentication Challenge

The Speaker has sent a Security Change Direction (SECD) command and must wait for Authentication Challenge (AUCH) from the Listener.

WF_AURP   Wait for Authentication Response

The Speaker has sent an Authentication Challenge (AUCH) command and must wait for Authentication Response (AURP) from the Listener.

9.4 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_NERP_RQ    F_ABORT_RQ     F_START_FILE_RS(-)   F_CLOSE_FILE_RS(-)
F_CD_RQ      F_RELEASE_RQ   F_RTR_RS
```

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
NERP	SECD	AUCH	AURP			

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.

9.5 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_NERP_IND	F_RELEASE_IND	F_DATA_CF	F_RTR_CF

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
NERP	SECD	AUCH	AURP			

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.

9.7 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 Data 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 use 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.
Caller	Yes/No	This entity initiated the ODETTE-FTP session.
Authentication	Yes/No	Secure authentication in use as agreed
Challenge	Binary	Random challenge

9.8 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 Data Exchange Buffer size supported.
Max-window	0 < Int < 1000	Local maximum credit value.
Cap-restart	Yes/No	Restart supported ?
Cap-logic	0, 1, 2	0 = does not support special logic 1 = supports special logic 2 = needs special logic

9.9 Session Connection State Table

9.9.1 State Table

```
o-----------------------------------------------------------o
|   | Other States                                          |
|   |--------------------------------------------------o    |
|   | WF_SECD                                          |    |
|   |----------------------------------------------o   |    |
|   | WF_AURP                                      |   |    |
|   |------------------------------------------o   |   |    |
```

```
|   | WF_AUCH                                  |   |   |   |
|   |--------------------------------------o   |   |   |   |
| S | A_WF_CONRS                           |   |   |   |   |
|   |----------------------------------o   |   |   |   |   |
| T | A_NC_ONLY                        |   |   |   |   |   |
|   |------------------------------o   |   |   |   |   |   |
| A | I_WF_SSID                    |   |   |   |   |   |   |
|   |--------------------------o   |   |   |   |   |   |   |
| T | I_WF_RM                  |   |   |   |   |   |   |   |
|   |----------------------o   |   |   |   |   |   |   |   |
| E | I_WF_NC              |   |   |   |   |   |   |   |   |
|   |------------------o   |   |   |   |   |   |   |   |   |
|   | IDLE             |   |   |   |   |   |   |   |   |   |
|==================o---+---+---+---+---+---+---+---+---+---|
|   | F_CONNECT_RQ | A | X | X | X | X | X | X | X | X | X |
|   |--------------+---+---+---+---+---+---+---+---+---+---|
| E | N_CON_CF     | X | C | X | X | X | X | X | X | X | X |
|   |--------------+---+---+---+---+---+---+---+---+---+---|
| V | SSRM         | X | X | H | X | X | X | L | L | L | X |
|   |--------------+---+---+---+---+---+---+---+---+---+---|
| E | SSID         | X | X | X | D | E | F | L | L | L | F |
|   |--------------+---+---+---+---+---+---+---+---+---+---|
| N | N_CON_IND    | B | X | X | X | X | X | X | X | X | X |
|   |--------------+---+---+---+---+---+---+---+---+---+---|
| T | F_CONNECT_RS | X | U | U | U | U | G | X | X | X | U |
|   |--------------+---+---+---+---+---+---+---+---+---+---|
|   | ESID         | X | X | X | F | X | X | F | F | F | X |
|   |--------------+---+---+---+---+---+---+---+---+---+---|
|   | AUCH         | X | X | U | U | X | X | I | L | L | U |
|   |--------------+---+---+---+---+---+---+---+---+---+---|
|   | AURP         | X | X | U | U | X | X | L | K | L | U |
|   |--------------+---+---+---+---+---+---+---+---+---+---|
|   | SECD         | X | X | U | U | X | X | L | L | J | U |
o----------------------------------------------------------o
```

9.9.2 Transition Table

```
 I | Predicate    Actions     Output Events            Next State
===o=============================================================
 A | P1:                      F_ABORT_IND              IDLE
   | !P1:            1,2      N_CON_RQ                 I_WF_NC
---+-------------------------------------------------------------
 B | P3:                      N_DISC_RQ                IDLE
   | !P3:            2        N_CON_RS
   |                          SSRM                     A_NC_ONLY
---+-------------------------------------------------------------
 C |                 4,2                               I_WF_RM
---+-------------------------------------------------------------
 D | P2 & P8 & P11:  4,2,5    SECD                     WF_AUCH
   | P2 & P8 & !P11: 4,2,5    F_CONNECT_CF             IDLESP
   | P2 & !P8:       4,2      ESID(R=12)
```

```
   |                          F_ABORT_IND(R,AO=L)      WF_NDISC
   | else:           4,2      ESID(R=10)
   |                          F_ABORT_IND(R,AO=L)      WF_NDISC
---+-----------------------------------------------------------------
 E | P4:             4        N_DISC_RQ                IDLE
   | !P4:            4,2      F_CONNECT_IND            A_WF_CONRS
---+-----------------------------------------------------------------
 F |                 4        F_ABORT_IND
   |                          N_DISC_RQ                IDLE
---+-----------------------------------------------------------------
 G | P2 &  P9 & P10: 4,2,5    SSID                     WF_SECD
   | P2 & !P9 & P10: 4,2,5    SSID                     IDLELI
   | !P10:           4,2      ESID(R=12)
   |                          F_ABORT_IND(R,AO=L)      WF_NDISC
   | else:           4,2      ESID(R=10)
   |                          F_ABORT_IND(R,AO=L)      WF_NDISC
---+-----------------------------------------------------------------
 H |                 4,2,3    SSID                     I_WF_SSID
---+-----------------------------------------------------------------
 I | P5:             4,2      AURP                     WF_SECD
   | !P5:            4,2      AURP                     IDLELI
---+-----------------------------------------------------------------
 J |                 4,2      AUCH                     WF_AURP
---+-----------------------------------------------------------------
 K | P6:             4,2      F_CONNECT_CF             IDLESP
   | P7:             4,2      SECD                     WF_AUCH
   | else:           4,2      ESID(R=11)
   |                          F_ABORT_IND(R,AO=L)      WF_NDISC
---+-----------------------------------------------------------------
 L |                 4,2      ESID(R=02)
   |                          F_ABORT_IND(R,AO=L)      WF_NDISC
---+-----------------------------------------------------------------
```

9.9.3 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:  SSID negotiation is successful
                  ( for these, Buf-size, Restart, Compression, Mode,
                    Special logic and Window, compare the inbound SSID
                    with the local constants to set the local variables.
                    Any incompatibilities result in failure of the
                    negotiation. )

   Predicate P3:  C.Cap-init = Initiator
```

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

   Predicate P5:  V.Caller = Yes

   Predicate P6:  (V.Caller = Yes) AND
                   (AURP.Signature verifies with V.Challenge)

   Predicate P7:  (V.Caller = No)  AND
                   (AURP.Signature verifies with V.Challenge)

   Predicate P8:  V.Authentication = I.SSID.Authentication

   Predicate P9:  I.F_CONNECT_RS.Authentication = Yes

  Predicate P10:  O.F_CONNECT_IND.Authentication =
                   I.F_CONNECT_RS.Authentication

  Predicate P11:  V.Authentication = Yes

       Action 1:  Set V.Mode from (C.Cap-mode, I.F_CONNECT_RQ.Mode)
                  Set V.Pswd, V.Id, V.Restart and
                   V.Authentication from I.F_CONNECT_RQ
                  Set V.Buf-size = C.Max-buf-size
                  Set V.Compression = C.Cap-compression
                  Set V.Caller = Yes
                  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, V.Authentication = from SSID

       Action 6:  Set V.Challenge = A random number unique to
                   the session
```

9.10 Error and Abort State Table

9.10.1 State Table

```
   o---------------------------------------o
   |   | Other States                      |
   | S |------------------------------o    |
   | T | WF_NDISC                     |    |
   | A |--------------------------o   |    |
   | T | I_WF_NC                  |   |    |
   | E |----------------------o   |   |    |
   |   | 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 |
o-------------------------------------o
```

9.10.2 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

9.10.3 Predicates and Actions.

Action 1: Stop inactivity timer

Action 2: Start inactivity timer

9.11 Speaker State Table 1

9.11.1 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

```
o--------------------------------------------------------------------o
| | Other States                                                     |
| |--------------------------------------------------------------o   |
| | WF_NDISC                                                     |   |
| |----------------------------------------------------------o   |   |
| | OPOWFC                                                   |   |   |
| |------------------------------------------------------o   |   |   |
| | OPO                                                  |   |   |   |
|S|--------------------------------------------------o   |   |   |   |
| | OPOP                                             |   |   |   |   |
|T|----------------------------------------------o   |   |   |   |   |
| | CDSTWFCD                                     |   |   |   |   |   |
|A|------------------------------------------o   |   |   |   |   |   |
| | SFSTWFCD                                 |   |   |   |   |   |   |
|T|--------------------------------------o   |   |   |   |   |   |   |
| | NRSTWFCD                             |   |   |   |   |   |   |   |
|E|----------------------------------o   |   |   |   |   |   |   |   |
| | ERSTWFCD                         |   |   |   |   |   |   |   |   |
| |------------------------------o   |   |   |   |   |   |   |   |   |
| | WF_CD                        |   |   |   |   |   |   |   |   |   |
| |--------------------------o   |   |   |   |   |   |   |   |   |   |
| | WF_RTR                   |   |   |   |   |   |   |   |   |   |   |
| |----------------------o   |   |   |   |   |   |   |   |   |   |   |
| | IDLESPCD             |   |   |   |   |   |   |   |   |   |   |   |
| |------------------o   |   |   |   |   |   |   |   |   |   |   |   |
| | IDLESP           |   |   |   |   |   |   |   |   |   |   |   |   |
|=+==============o---+---+---+---+---+---+---+---+---+---+---+---+---|
| | F_EERP_RQ    | A | A | W | F | W | W | U | U | U | U | U | U | U |
| |--------------+---+---+---+---+---+---+---+---+---+---+---+---+---|
| | F_NERP_RQ    | Y | Y | W | Z | W | W | U | U | U | U | U | U | U |
| |--------------+---+---+---+---+---+---+---+---+---+---+---+---+---|
| | F_START_     | B | B | W | G | W | W | U | U | U | U | U | X | U |
| |   FILE_RQ    |   |   |   |   |   |   |   |   |   |   |   |   |   |
| |--------------+---+---+---+---+---+---+---+---+---+---+---+---+---|
| | SFPA         | C | C | C | C | C | C | C | C | K | C | C | S | C |
| |--------------+---+---+---+---+---+---+---+---+---+---+---+---+---|
|E| SFNA         | C | C | C | C | C | C | C | C | L | C | C | S | C |
| |--------------+---+---+---+---+---+---+---+---+---+---+---+---+---|
|V| CD           | C | C | C | H | R | Z1| I | J | C | C | C | S | C |
| |--------------+---+---+---+---+---+---+---+---+---+---+---+---+---|
|E| F_DATA_RQ    | U | U | U | U | U | U | U | U | U | M | U | S | U |
| |--------------+---+---+---+---+---+---+---+---+---+---+---+---+---|
|N| CDT          | C | C | C | C | C | C | C | C | C | P | O | S | C |
| |--------------+---+---+---+---+---+---+---+---+---+---+---+---+---|
|T| F_CD_RQ      | D | U | W | T | W | W | U | U | U | U | U | X | U |
| |--------------+---+---+---+---+---+---+---+---+---+---+---+---+---|
| | F_REL_RQ(Ok) | U | E | U | U | U | U | U | U | U | U | U | X | U |
| |--------------+---+---+---+---+---+---+---+---+---+---+---+---+---|
| | F_REL_RQ(Err)| Q | Q | Q | Q | Q | Q | Q | Q | Q | Q | Q | S | Q |
| |--------------+---+---+---+---+---+---+---+---+---+---+---+---+---|
| | RTR          | C | C | N | C | C | C | C | C | C | C | C | S | C |
```

```
 o-------------------------------------------------------------------o
```

9.11.2 Transition Table

```
     I | Predicate    Actions      Output Events              Next State
    ===o==================================================================
     A | P5:          1,2,3,18     EERP                       WF_RTR
       | !P5:         1,2,3        EERP                       WF_RTR
    ---+------------------------------------------------------------------
     B | P1:                                                  UE
       | !P1:         1,2,5        SFID                       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:                                                  UE
       | !P1:         6                                       SFSTWFCD
    ---+------------------------------------------------------------------
     H |              1,2                                     IDLESP
    ---+------------------------------------------------------------------
     I |              1,2,10       SFID                       OPOP
    ---+------------------------------------------------------------------
     J |              1,2          CD                         IDLELICD
    ---+------------------------------------------------------------------
     K | P2:          1,2          ESID(R=02)
       |                           F_ABORT_IND(R,AO=L)        WF_NDISC
       | !P2:         1,2,7,12     F_START_FILE_CF(+)         OPO
    ---+------------------------------------------------------------------
     L |              1,2,8        F_START_FILE_CF(-)         IDLESP
    ---+------------------------------------------------------------------
     M | P3:          1,2,11,13    DATA                       OPOWFC
       | !P3:         1,2,11,13    DATA
       |                           F_DATA_CF                  OPO
    ---+------------------------------------------------------------------
     N |                           F_RTR_CF                   IDLESP
    ---+------------------------------------------------------------------
     O |              12           F_DATA_CF                  OPO
    ---+------------------------------------------------------------------
     P | Protocol     1,2          ESID(R=02)
       | Error                     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
W			User Error - Note 1	UE
X			Error	
Y	P4 & P5:	1,2,15,18	NERP	WF_RTR
	!P4 & !P5:	1,2,15,14	NERP	WF_RTR
	P4 & !P5:	1,2,15	NERP	WF_RTR
	!P4 & P5:	1,2,15,14,18	NERP	WF_RTR
Z		16		NRSTWFCD
Z1	P4:	1,2,17	NERP	WF_RTR
	!P4:	1,2,17,14	NERP	WF_RTR

9.11.3 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.

Predicate P4: No special logic is in use

Predicate P5: Signed EERP/NERP requested

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

Action 14: Activate CRC-calculus function. Wrap Exchange buffer
           in special logic

Action 15: Build a NERP from F_NERP_RQ

Action 16: Store F_NERP_RQ in V.Req-buf

Action 17: Build NERP from F_NERP_RQ stored in V.Req-buf

Action 18: Sign the contents of NERP/EERP

Note 1: Whether 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).

9.12 Speaker State Table 2

9.12.1 State Table

```
o---------------------------------o
| S | CLOP                        |
| T |-------------------------o   |
| A | OPOWFC                  |   |
| T |---------------------o   |   |
| E | OPO                 |   |   |
|=====================o---+---+---|
| E | F_CLOSE_FILE_RQ | A | E | U |
| V |-----------------+---+---+---|
```

```
| E | EFPA             | B | B | C |
| N |-----------------+---+---+---|
| T | EFNA             | B | B | D |
o---------------------------------o
```

9.12.2 Transition Table

```
 I | Predicate    Actions     Output Events            Next State
===o=====================================================================
 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
   | !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
-------------------------------------------------------------------------
```

9.12.3 Predicates and Actions.

Predicate P1: (I.EFPA.CD-Request = Yes)

Predicate P2: No special logic is in use

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

Action 8: Wrap Exchange buffer in special logic

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.

9.13 Listener State Table

9.13.1 State Table

```
o---------------------------------------------o
|   | RTRP                                    |
|   |-------------------------------------o   |
|   | CLIP                                |   |
|   |---------------------------------o   |   |
|   | OPI                             |   |   |
| S |-----------------------------o   |   |   |
| T | OPIP                        |   |   |   |
| A |-------------------------o   |   |   |   |
| T | IDLELICD                |   |   |   |   |
| E |---------------------o   |   |   |   |   |
|   | IDLELI              |   |   |   |   |   |
|=====================o---+---+---+---+---+---+
|   | SFID            | A | A | B | B | B | B |
|   |-----------------+---+---+---+---+---+---+
| E | DATA            | B | B | B | I | B | B |
| V |-----------------+---+---+---+---+---+---+
| E | EFID            | B | B | B | J | B | B |
| N |-----------------+---+---+---+---+---+---+
| T | F_START_FILE_RS | U | U | H | U | U | U |
|   |-----------------+---+---+---+---+---+---+
|   | F_CLOSE_FILE_RS | U | U | U | U | K | U |
|   |-----------------+---+---+---+---+---+---+
|   | CD              | C | B | B | B | B | B |
|   |-----------------+---+---+---+---+---+---+
|   | ESID R=Normal   | D | F | D | D | D | D |
|   |-----------------+---+---+---+---+---+---+
|   | ESID R=Error    | D | D | D | D | D | D |
|   |-----------------+---+---+---+---+---+---+
|   | EERP            | E | E | B | B | B | B |
|   |-----------------+---+---+---+---+---+---+
|   | NERP            | L | L | B | B | B | B |
|   |-----------------+---+---+---+---+---+---+
|   | F_RTR_RS        | U | U | U | U | U | M |
o---------------------------------------------o
```

9.13.2 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
   | !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 |                    1,2,4      F_EERP_IND                RTRP
---+-----------------------------------------------------------------
 F |                    1          F_RELEASE_IND
   |                               N_DISC_RQ                 IDLE
---+-----------------------------------------------------------------
 H |  P4:                          User Error                UE
   |  P2 & !P4 & !P5:   1,2,8      SFPA                      OPI
   | !P2 & !P4 & !P5:   1,2        SFNA                      IDLELI
   |  P2 & !P4 &  P5:   1,2,5,8    SFPA                      OPI
   | !P2 & !P4 &  P5:   1,2,5      SFNA                      IDLELI
---+-----------------------------------------------------------------
 I | P6:                1,2        ESID(R=02)
   |                               F_ABORT_IND(R,A0=L)       WF_NDISC
   | !P5 & !P6 & !P7:   1,2,7      F_DATA_IND (See Note 1)   OPI
   | !P5 & !P6 &  P7:   1,2,8      F_DATA_IND
   |                               CDT (See Note 1)          OPI
   |  P5 & !P6 &  P8:   1,2        ESID(R=07)
   |                               F_ABORT_IND(R,A0=L)       WF_NDISC
   |  P5 & !P6 & !P7 :  1,2,6,7    F_DATA_IND (See Note 1)   OPI
   |   & !P8
   |  P5 & !P6 &  P7 :  1,2,5,6,8  F_DATA_IND                OPI
   |   & !P8                       CDT (See Note 1)
---+-----------------------------------------------------------------
 J |                    1,2        F_CLOSE_FILE_IND          CLIP
---+-----------------------------------------------------------------
 K |  P2 &  P3 & !P5:   1,2        EFPA(CD-Req)              WF_CD
   |  P2 & !P3 & !P5:   1,2        EFPA(no CD)               IDLELI
   | !P2 & !P5:         1,2        EFNA                      IDLELI
   |  P2 & !P3 & P5:    1,2,5      EFPA(no CD)               IDLELI
   | !P2 &  P5:         1,2,5      EFNA                      IDLELI
   |  P2 &  P3 & P5:    1,2,5      EFPA(CD-Req)              WF_CD
---+-----------------------------------------------------------------
 L |                    1,2,10     F_NERP_IND                RTRP
---+-----------------------------------------------------------------
 M |                    1,2        RTR                       IDLELI
---+-----------------------------------------------------------------
 U |                               User Error                UE
--------------------------------------------------------------------
```

9.13.3 Predicates and Actions.

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

(V.Mode = Sender-only)

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: Special logic is used

Predicate P6: V.Credit_L - 1 < 0

Note: Protocol Error because the Speaker has exceeded its available transmission credit.

Predicate P7: V.Credit_L - 1 = 0

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

Predicate P8: The calculus of the received CRC indicates an error

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: Add special logic header to the command to be sent to the speaker

Action 6: Suppress the special logic header from the data buffer before giving it to the user.

Action 7: V.Credit_L = V.Credit_L - 1

Action 8: V.Credit_L = V.Window

Action 10: Build F_NERP_IND from I.NERP

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 to receive data.
2. The number of buffers available to ODETTE-FTP.

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

9.14 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                                            // invalid restart
   Actions:   Stop inactivity timer,         // reset timer
              Start inactivity 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
```

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.

10. Miscellaneous

10.1 Algorithm Choice

The choice of algorithms to use for security or compression between partners is for bilateral agreement outside of the ODETTE-FTP.

10.2 Cryptographic Algorithms

The algorithms for symmetric and asymmetric cryptography and hashing are represented by a coded value, the cipher suite:

Cipher Suite	Symmetric	Asymmetric	Hashing
01	3DES_EDE_CBC_3KEY	RSA_PKCS1_15	SHA-1
02	AES_256_CBC	RSA_PKCS1_15	SHA-1

Support of all cipher suites listed here is mandatory.

The certificates used must be [X.509] certificates.

TripleDES is using Cipher Block Chaining mode (CBC) for added security and uses the EDE (Encryption Decryption Encryption) process with 3 different 64 bit keys.

RSA padding is as defined in [PKCS #1].

AES is using a 256 bit key in Cipher Block Chaining mode (CBC).

An extended list of optional cipher suites may be used (Section 10.2) but there is no guarantee that two communicating ODETTE-FTP entities would both support these optional cipher suites.

10.2 Protocol Extensions

The algorithms and file enveloping formats available in ODETTE-FTP may be extended outside of this document.

A free list of optional extensions authorised for use as part of ODETTE-FTP is available from ODETTE International Ltd and on their website at http://www.odette.org

10.3 Certificate Services

Certificates and certificate revocation lists may be exchanged as [CMS] enveloped files. It is therefore valid to exchange a [CMS] file that is neither encrypted, compressed or signed. It is an application implementation issue to determine the correct course of action on receipt of such a file.

10.4 Security Considerations

ODETTE-FTP security requires the use of [X.509] certificates. If no security options are agreed for use, the send and receive passwords are sent in plain text. Whilst this is acceptable over X.25 and ISDN networks, this is a risky practice over insecure public networks such as the Internet.

All, some or none of the security options available in ODETTE-FTP may be used. No recommendations for the use of these options are provided in this specification. Whilst use of the highest strength encryption algorithms may seem admirable there is often a performance tradeoff to be made, and signing all files and acknowledgements has potential legal implications that should be considered.

It should be noted that whilst the security measures ensure that an ODETTE-FTP partner is authenticated, it does not necessarily mean that the partner is authorised. Having proven the identity of a partner, it is an application issue to decide whether that partner is allowed to connect or exchange files.

Extracted from [RFC 3850]:

"When processing certificates, there are many situations where the processing might fail. Because the processing may be done by a user agent, a security gateway, or other program, there is no single way to handle such failures. Just because the methods to handle the failures have not been listed, however, the reader should not assume that they are not important. The opposite is true: if a certificate is not provably valid and associated with the message, the processing software should take immediate and noticeable steps to inform the end user about it.

Some of the many situations in which signature and certificate checking might fail include the following:

No certificate chain leads to a trusted CA
No ability to check the Certificate Revocation List (CRL) for a certificate
An invalid CRL was received
The CRL being checked is expired
The certificate is expired
The certificate has been revoked

There are certainly other instances where a certificate may be invalid, and it is the responsibility of the processing software to check them all thoroughly, and to decide what to do if the check fails. See RFC 3280 for additional information on certificate path validation."

The push / pull nature of ODETTE-FTP means that a party can make an outbound connection from behind a firewall to another party and exchange files in both directions. There is no need for both partners to open ports on their firewalls to allow incoming connections - only one party needs to allow incoming connections.

See Section 1.7 for a discussion of the benefits of session security [TLS] versus file security.

Appendix A. Virtual File Mapping Example

This example demonstrates the mapping of a Virtual File into a sequence of ODETTE-FTP Data Exchange Buffers.

Each line in this extract from 'The Rime of the Ancient Mariner' by Coleridge [RIME] is considered to be a separate record in a file containing variable length records, that is being transmitted as a V Format file.

```
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 opended 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, while the two lines below give the hexadecimal value.

Note 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

```
D.It is an ancient Mariner,.And he stoppeth one of three.."By th
4947267266266666672467666729466266277677676266626627676662A247276
499409301E01E395E40D129E52CD1E4085034F005480FE50F6048255EB229048

y long grey beard and glittering eye,."Now wherefore stopp'st th
7266662676726667626662666776766626762A24672766766676277677277276
```

```
  90CFE70725902512401E407C944529E70595C12EF70785256F25034F00734048

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

   of kin;."The guests are met, the feast is set :."May'st hear th
  26626663A25662676777267626672276626667726727672302467277266672 76
  0F60B9EB72485075534301250D54C0485065134093035 40AF2D1973408512048

Exchange buffer  2

  D.e merry din.".He holds him with his skinny hand,."There was a
  486266777266622A462666672666276762667276666726666292566762767262
  4D50D5229049EE228508FC43089D07948089303B9EE9081E4CD2485250713010

  ship," quoth he.."Hold off | unhand me, grey-beard loon |".Eftso
  7667222776762662A24666266622276666626662267672666762666622294677 6
  3890C2015F48085E928FC40F660105E81E40D5C07259D251240CFFE012B5643F

  ons his hand dropt he..He holds them with his glittering eye -.T
  6672667266662676772662A462666672766627676266726667767666266762295
  FE30893081E4042F04085E78508FC430485D07948089307C944529E705950DE4

  he Wedding-Guest stood still,.And listens like a three years' ch
  6625666666624767727766627766622466266776672666662627676627667722 66
  85075449E7D75534034FF40349CCC21E40C9345E30C9B5010482550951237038

Exchange buffer  3

  D.ild :.The Mariner hath his will..The Wedding-Guest sat on a st
  4866623956624676667266762667276662A566256666662476772767266262 77
  459C40AA4850D129E52081480893079CCE2485075449E7D7553403140FE01034

  one :.He cannot chuse but hear ;.And thus spake on that ancient
  666239462666666726677626772666723A4662767727766626627667266666672
  FE50AA85031EEF40385350254085120B31E4048530301B50FE0481401E395E40

  man,.The bright-eyes Mariner..The ship was cheered, the harbour
  66629566267666726767246766672A566276672767266667662276626676677 2
  D1EC84850229784D59530D129E52EA4850389007130385525 4C048508122F520

  cleared,.Merrily did we drop.Below the kirk, below the hill,.Bel
  666676629467766726662762676794666727662667622666672766266662046 6
  3C51254C3D5229C90494075042F0F25CF704850B92BC025CF70485089CCC325C

Exchange buffer  4

  D.ow the light-house top.
  4967276626666726677627672
  47F704850C9784D8F53504F0E
```

Appendix B. ISO 646 Character Subset

BIT 4	3	2	1		0	1	2	3	4	5	6	7
				BIT 7	0	0	0	0	1	1	1	1
				BIT 6	0	0	1	1	0	0	1	1
				BIT 5	0	1	0	1	0	1	0	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			

Appendix C. X.25 Specific Information

The International Standards Organisation (ISO) Open System

Interconnection (OSI) model is the basis for the ODETTE-FTP.

The ODETTE-FTP covers levels 4 to 7 and originally CCITT X.25 was the only recommended telecommunication protocol for OSI's layers 1, 2, 3.

ISO Reference Model :

```
+------------------------------+  <====  File Service
| Level-7  FTP    application  |
|------------------------------|
| 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         |
+------------------------------+
```

C.1 X.25 Addressing Restrictions

When an X.25 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.

C.2 Special Logic

The SSID field SSIDSPEC specifies whether special logic must be applied (Y (yes) or N (no)) to the data exchange buffer before the ODETTE-FTP moves the data into the NSDU (Network Service Data Unit) and passes control to the network service.

C.2.1 When special logic is not to be used

This logic is not applied to SSRM and SSID commands.

C.2.2 The need for "enveloping" exchange buffers

The "special-logic" was created in order to allow the use of ODETTE-FTP over asynchronous links. The "special-logic" could be needed to enable terminals to access an X.25 network via an asynchronous entry (through a PAD: Packet Assembly / Disassembly). The "special-logic" is not needed in case of a whole X.25 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:

```
        SSID                                       SSID
        -----------------------------------------------

        special-logic=yes --------------------->

           <-------------------------------------   special-logic=yes
                                               or
           <-------------------------------------   special-logic=no


        special-logic=no ---------------------->

           <-------------------------------------   special-logic=no
```

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.

```
       +-------------------------------+  <====  File Service
       | Level-7  FTP    application  |
       |-------------------------------|
       | Level-6  FTP    presentation |
       |-------------------------------|
       | Level-5  FTP    session      |
       |-------------------------------|
       | Level-4  FTP    transport    |
```

```
         |  SPECIAL LOGIC PROCESSING    |
         |------------------------------|  <====  Network Service
         | Level-3         X.25         |
         |------------------------------|
         | Level-2         X.25         |
         |------------------------------|
         | Level-1         X.25         |
         +------------------------------+
```

C.2.3 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.

C.2.4 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:

```
+-------------------------------------------------------------------+
| S | B |                                                 | B | C |
| T | S |         COMPLETE EXCHANGE BUFFER (CEB)          | C | / |
| X | N |                                                 | S | R |
+-------------------------------------------------------------------+
  A   A                                                     A   A
  |   |                                                     |   |
  |   +-------------  Block sequence number                 |   |
  |                                                         |   |
  +-----------------  Synchronization character             |   |
                                                            |   |
                      Block checksum  -----------------------+   |
                                                                |
                      Delineation character  --------------------+
```

The envelope is initialised with an 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 it's 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 initialise 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.

C.2.5 Error recovery

C.2.5.1 Mechanism

The error correction scheme is implemented by the definition of three Timers and the use of an 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.

A 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 a 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.

C.2.5.2 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 while 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.

C.2.5.3 Types of error

Data corruption when it occurs can be categorised in 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.

```
    (A)                                      (B)

Dropped Start of Text (STX)

      +-------------------------+
      |   | B |         | B | C |
 -----|   | S |  CEB    | C | / |----->  Not sync
      |   | N |         | S | R |
      +-------------------------+

               +-------+
               | N | C |
         <-----| A | / |-----            Not sync
               | K | R |
               +-------+

Exchange Buffer Resent

      +-------------------------+
      | S | B |         | B | C |
 -----| T | S |  CEB    | C | / |----->  Sync
      | X | N |         | S | R |
      +-------------------------+
```

(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

      +-------------------------+
      | S | B |         | B |   |
 -----| T | S |  CEB    | C |   |----->  No activity
      | X | N |         | S |   |
      +-------------------------+

               +-------+
               | N | C |                 T2
         <-----| A | / |-----            Timed out
```

```
                | K | R |
                +-------+
```

Exchange Buffer Resent

```
       +--------------------------+
       | S | B |         | B | C |
  -----| T | S |  CEB    | C | / |----->  Sync
       | X | N |         | S | R |
       +--------------------------+
```

(3) BAD DATA
(4) BAD BCS (BLOCK CHECK SUM)

In this situation, the receiver is unable to tell 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

```
       +--------------------------+
       | S | B |         | B | C |          Bad data
  -----| T | S |  "%!    | C | / |----->  detected
       | X | N |         | S | R |
       +--------------------------+

                +-------+
                | N | C |
          <-----| A | / |-----            Discard Block
                | K | R |
                +-------+
```

Exchange Buffer Resent

```
       +--------------------------+
       | S | B |         | B | C |
  -----| T | S |  CEB    | C | / |----->  Data OK
       | X | N |         | S | R |
       +--------------------------+
```

(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 in a protocol violation.

```
Example protocol sequence:

    (A)                                       (B)

Exchange Buffer Being Sent

      +-------------------------+
      | S |   |         | B | C |        Expecting
 -----| T | 0 |  EERP   | C | / |----->  BSN=0
      | X |   |         | S | R |        Transmission
      +-------------------------+

Exchange Buffer Being Sent

      +-------------------------+
      | S |   |         | B | C |        Response to
 <----| T | 0 |  RTR    | C | / |-----   Previous
      | X |   |         | S | R |        Block
      +-------------------------+

Exchange Buffer Being Sent

      +-------------------------+        Expecting
      | S |   |         | B | C |        BSN=1 (Block
 -----| T | 1 |  SFID   | C | / |- // -> lost in
      | X |   |         | S | R |        Transmission)
      +-------------------------+        T1 Timed Out

Exchange Buffer Being Sent

      +-------------------------+
      | S |   |         | B | C |        Send last
 <----| T | 0 |  RTR    | C | / |-----   Block
      | X |   |         | S | R |        again
      +-------------------------+

Discard Block
and start
Timer T1

T1 Timed Out

Exchange Buffer Resent

      +-------------------------+
      | S |   |         | B | C |        Expecting
 -----| T | 1 |  SFID   | C | / |----->  BSN=1
      | X |   |         | S | R |        Block OK
      +-------------------------+

Exchange Buffer Being Sent
```

```
      +-------------------------+
      | S |   |         | B | C |          Response
 <----| T | 1 |  SFPA   | C | / |-----    BSN=1
      | X |   |         | S | R |          Block OK
      +-------------------------+
```

Exchange Buffer Being Sent

```
      +-------------------------+
      | S |   |         | B | C |
 -----| T | 2 |  DATA   | C | / |----->  Data OK
      | X |   |         | S | R |
      +-------------------------+
```

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

C.2.6 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.

C.2.7 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 NSDU
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 if was plus zero (i.e. 0).

ALGORITHM FOR GENERATING CHECKSUM PARAMETERS :

. Set up the complete NSDU with the value of the checksum parameter field set to zero.

. Initialise 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.

C.2.8 Algorithm for checking checksum parameters

. Initialise parameters C0 and C1 to zero.

. Process each octet of NSDU 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.

C.2.9 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:

C.3 PAD Parameter profile

Before an (ODETTE-FTP) asynchronous entity --> Modem--> PAD--> (ODETTE-FTP) 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 ODETTE-FTP 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)
- Parameters 1 - 12 are mandatory and available internationally.
- Parameters 13 - 22 may be available on certain networks 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 ODETTE-FTP special logic option may be impossible on some PADs because of none support of some of the parameters (13 - 22). (If the PAD is supporting parity check (21) by default, ODETTE-FTP 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 ODETTE-FTP monitor function.
- PAD services implemented.
- Packet mode DTE entity ODETTE-FTP monitor function.

Appendix D. OFTP X.25 Over ISDN Recommendation

This appendix describes the recommendation of ODETTE Group 4 (1) for the use of OFTP (2) over X.25 over ISDN.

This document offers an introductory overview of a technical subject. It is structured to contain the ODETTE recommendation, together with introductory information for the person not familiar with ISDN and also notes on the issues associated with the implementation of the recommendation.

The first section provides the detailed ODETTE recommendation which is followed by a general discussion. If you are not familiar with the terminology, please read the subsequent sections first.

How far an existing X.25 Line adapter may be replaced by an ISDN line adapter in an installation depends on the opportunities in view of connections (X.25 or ISDN) of the involved partners for file transfer.

Companies, which keep many connections to external partners (for example car manufacturing companies), may use the OFTP file transfer in view of compatibility, which must always be considered, anyway only in parallel to the X.25 network.

It is not the aim of this recommendation, to remove the OFTP file transfer generally from the X.25 network to the ISDN network. This will not always be possible for international connections because of technical reasons, and this does not always make sense for connections with a low size of data to be transmitted.

Certainly the use of ISDN, when exchanging a high volume of data (for example CAD/CAM files), is very much cheaper than the use of an X.25 network. For such cases this recommendation shall provide a cost effective possibility for file transfer.

(1) ODETTE Group 4 is responsible for the specification of Telecommunications standards and recommendations for use within the Automotive Industry.

(2) OFTP (ODETTE File Transfer Protocol) is the communications standard specified by ODETTE Group 4 designed for the transfer of both EDI and non-EDI data.

Contents

D.1 ODETTE ISDN Recommendation

X.25:	Level 2 Protocol	ISO 7776
	Level 3 Protocol	ISO 8208
	Packet Size	128
	Level 2 Window Size	7
	Level 3 Window Size	7
	First LCN	1
	Number of LCNs	1
	Facilities	Window Size and Packet Size negotiation shall be supported by everybody. Call User Data should not be required.

Calling NUA	Optionally provided by the call initiator.
Called NUA	Should be set to a value where the last 'n' digits can be specified by the called party.

ISDN: Apart from requesting a 64K unrestricted digital call, no ISDN features shall be required.

Timeout control: To avoid connections (B-Channels) within the circuit switched ISDN network remaining active but unused for a long time, the adapter should include a timeout control.

An ISDN connection (B-channel) should be released if no X.25 packets have been transmitted on this connection for a longer time. For flexibility a variable user definable timer should be incorporated into the adapter.

In the event of a timeout situation the adapter has to release the ISDN connection and notify the local OFTP by the transmission of a clear packet.

The pages that follow are informational and do not form part of this recommendation

D.2 Introduction to ISDN

The use of digital encoding techniques over such high quality, error free, backbone networks has allowed the PTTs to offer high bandwidths to the end user. The service is named ISDN (Integrated Services Digital Network).

The increasing need to transfer larger volumes of EDI data, in particular CAD/CAM drawings, has focused attention upon high speed, low cost, communication. The traditional X.25 over a Packet Switched Data Network (PSDN) has been a good general purpose communications subsystem. Unfortunately its cost and transfer speed make PSDN expensive for the new requirement.

X.25 over the new ISDN provides both, the transfer speed and cost benefits to satisfy the new requirements.

Terminology: For us to make sense of ISDN and X.25 it is important that we use definitions precisely and avoid the abuses of the past.

ISDN: Integrated Services Digital Network

X.25: X.25 is a communications protocol. It defines the structure of data packets that comprise the protocol and the manner in which they are used.

PSDN: A PSDN (Packet Switching Data Network) is a network over which the X.25 protocol is operated.

PSPDN: A PSPDN (Packet Switching Public Data Network) are PSDNs operated by the PTTs. PSPDNs are given Trade Names, such as PSS in the UK, Datex-P in Germany and Transpac in France.

BRI: Basic Rate Interface, also known as Basic Rate Access, defines an ISDN facility with 2 x 64K B-Channels.

PRI: Primary Rate Interface, also known as Primary Rate Access, defines an ISDN facility with 30 x 64K B-Channels.

Channels: ISDN is typically brought into a consumer's premises using a twisted pair of wire. Over this wire data can be transmitted in frequency bands. These frequency bands are allocated as channels.

B Channels: The B Channels are the data channels and operate at 64Kb. The two end users of a connection will communicate over a B Channel.

D Channel: Signalling on ISDN is performed over the D Channel. Signalling is used to setup and release connections on the B channels. In some countries the D channel can also be used for limited X.25 access to the PTTs PSDN.

The D channel operates at the lower speed of 16Kb as it is normally used only at the beginning and end of a connection.

```
                        Bandwidth Allocation:
    2 Wire                 B2 - 64 Kbit
    Twisted Pair           B1 - 64 Kbit
                        D Channel - 16 Kbit
```

The standard for the operation of the D channel is called ETSI and is used in most European countries. However some countries that started the introduction very early used proprietary standards e.g.

1TR6 Used in Germany
BTNR Used in UK

Although there are D channel variations, this will not affect communications over the B channels as the communication over the D channel is between the subscriber and the ISDN service provider.

However, the consumer's equipment must be able to handle the channel D signalling operated by the ISDN service provider and so there may be a problem of equipment availability and certification.

All the PTTs have committed to migrate to ETSI (3) and many are currently supporting both, their national variant and ETSI. It is advisable that in this situation the subscriber select the ETSI variant to avoid unnecessary equipment obsolescence.

(3) Also known as EURO-ISDN and as Q.931

Services: The high speed service is provided in two forms, Basic and Primary.

Basic: 2+D, the D 2B channel operates at 16 Kb. The Basic Rate access is normally provided to the subscriber over simple twisted pair cable.

Primary: 30B+D, the D channel operates at 64 Kb. Primary Rate access is normally provided to the subscriber over shielded coaxial cable. Note, that the bandwidth for Primary is 2.048 Mbit/s.

Protocols: The B channel is a binary channel and is transparent to the flow of data. Therefore all of the currently available protocols can operate over a B channel. The most common protocols are:

X.25: The X.25 protocol is a primary protocol for open computer to computer communication.

Passive Bus: It is possible to have an ISDN service enter a building and then have an 8 core cable laid within the building with multiple ISDN junction points, in the same way as one would have multiple telephone points (extensions) for a particular external telephone line.

Connection Setup

The adapter is responsible for analysing the outgoing X.25 call request and making an ISDN call to a derived ISDN address, establishing a new X.25 level-2 and level-3; then propagating the X.25 Call Request Packet.

Connection Termination

The termination phase of the X.25 call is made with a Clear Request and finalised with a Clear Confirmation. The recipient of the Clear Confirm should then closedown the ISDN connection.

The clear down of the ISDN connection should only be made if there are no other SVCs active on the ISDN connection; note that the usage of multiple simultaneous SVCs is only by virtue of bi-lateral agreement.

D.3 Equipment Types

There are a number of ways in which ISDN/X.25 access can be made.

Integrated Adapter

This is normally a PC based ISDN adapter inside a PC. It is normal in such an environment that the OFTP application has the ability to manipulate the ISDN and X.25 aspects of the session independently and therefore have complete control.

Equally important, is that the speed of communication between the adapter and the application are at PC BUS speeds. It is therefore more likely that the effective transmission speed will be nearer the 64K limit.

The other benefit of such a direct linkage, is that both 64K B channels may be used in parallel and both able to operate at 64Kb.

Elementary Terminal Adapter

In this scenario, the computer has an integral X.25 adapter communicating X.21 with a Terminal Adapter that fronts the ISDN network. This allows a host with a X.25 capability to interface to ISDN, normally on a one to one

The interface between the Terminal Adapter and the PC will typically only support one 64K B channel. This is obviously an inefficient usage of the ISDN service.

Because the linkage between the computer and the Terminal Adapter is only X.25, then some modification/configuration may be needed inside the Terminal Adapter when new users are added.

X.25 Switch

This solution is normally found inside the larger corporates where an internal X.25 network is operated or where dual X.25 and ISDN is required.

The main benefit of a switch is to support both PSDN and ISDN simultaneously. Also multiple X.21 lines may be implemented between the X.25 Switch and the computer.

This solution normally requires more effort to configure and may require obligations to be placed upon how incoming callers specify routing.

D.4 Implementation

Introduction

The adoption of ISDN as an additional sub-system to support OFTP communications has associated implementation problems which can be categorised as below:

X.25/ISDN Addressing
Making a call
Receiving a call
Logical Channel assignment
Facilities Negotiation
ISDN call attributes
Homologation Issues
Performance
Growth

X.25/ISDN Addressing

The original OFTP was designed to work over the X.25 networks provided by the PTTs (PSPDNs). The national X.25 networks were interconnected to provide a global X.25 network and a common addressing scheme was adopted by all. Although there were a few differences in addressing within a national network, the interface to other countries was quite rigid and normalised.

PSPDN Numbering

The addressing scheme adopted in X.25 is a 15 digit number (Network User Address, NUA) where the first three identify the country, the fourth digit identifies the network within the country and the remainder specify the individual subscriber plus an optional subaddress. In the UK where a full X.25 numbering scheme is adopted, a NUA is e.g. 234221200170; where 2342 is the DNIC (Data Network Identification Code) and 21200170 is the subscriber number.

ISDN Numbering

ISDN is an extension of the normal telephone system, consequently it adopts (or rather is) the same numbering scheme as the telephone system (PSTN).

The Numbering Conflict

The PSDN and PSTN numbering schemes are two totally different numbering schemes. There is no relationship between them. It is this conflict that is at the heart of the matter.

Making a Call

It is a consequence of PSDN and PSTN being based upon different and unconnected numbering schemes that the key problem arises.

For X.25 to work over ISDN, three main methods of addressing are available:

Un-mapped: The X.25 called NUA is used as the PSTN number. Thus an X.25 call to 0733394023 will result in a PSTN call to 0733394023 and the call request that consequently flows will also be to 0733394023.

Manipulated: The X.25 called NUA is manipulated by the subtraction and/or addition of digits to derive a resultant PSTN number. Thus 2394023 could be manipulated to derive a PSTN number of 00944733394023; where the prefix 2 is deleted and replaced by 00944733.

Mapped: The X.25 called NUA is used as a look-up into a table of PSTN numbers. Thus an X.25 call to 234221200170 could be mapped to and result in a PSTN call to 0733394023 and the call request that consequently flows will remain as 234221200170.

Un-mapped Calls

Un-mapped calls are where the host specified X.25 NUA is converted directly to the corresponding ISDN number.

Thus an X.25 call issued by the host to X.25 NUA 0733394023 will result in an ISDN call to the PSTN number 0733394023. After the call has been established, then HDLC/X.25 protocol setup will be established after which an X.25 call request will be transferred with the NUA 0733394023.

When a PSTN call is made, the number of digits in the called number vary depending upon the location of the called party.

When a number is called, it may be local, national or international.

local: 394023
national: 0733 394023
international: 009 44 733 394023

Depending upon where a call originates, the corresponding X.25 NUA in the call request packet will vary dramatically.

Such variation of X.25 NUA, in particular the changing prefix, can be difficult to be accommodated by X.25 routing logic in many products.

When an international PSTN call is being made, then it is likely that the PSTN number exceeds 15 digits, which is the maximum length of an X.25 NUA. Therefore, using un-mapped addressing may make some international calls impossible to make.

Manipulated Calls

The X.25 called NUA is manipulated by the subtraction and/or addition of digits to derive a resultant PSTN number.

Let us assume that by internal convention we have identified the prefix '2' to indicate an international ISDN call. Thus an X.25 call request of 244733394023 could be manipulated to derive a PSTN number of 00944733394023; where the prefix '2' is deleted and replaced by '009' (the international prefix).

The X.25 call NUA would typically be left in its un-manipulated state. As individual internal conventions vary, the X.25 call NUA will vary, in the case above it would be 244733394023, but another installation might have the convention where a prefix of '56' specifies the UK and so the NUA will be 56733394023 where the '56' is deleted and replaced with '00944' to derive the PSTN number.

Mapped Calls

The mapped method offers maximum flexibility in that:

The PSTN number can exceed 15 digits.

The X.25 NUA and PSTN number can be totally different.

The problem with mapped calls is administrative. IBM mainframes can't handle X.25 over ISDN at all, let alone support mapping. For the mainframe solution to work an external X.25/ISDN router box is required and it is the responsibility of the external box to provide any mapping necessary.

This means that any changes or addition of OFTP partners over ISDN will require access to the Computer room or special configuration equipment to change the tables inside the external X.25/ISDN router box.

Receiving Calls

We have seen from the previous section that the called X.25 NUA from an ISDN incoming call may vary considerably. If ISDN/X.25 is confined to a national boundary, then such variation will not be so great as most calls will have matching called X.25 NUA and PSTN numbers.

X.25 switches and X.25 adapters normally route/accept/reject calls based upon their X.25 called NUA. In particular, routing is made upon the X.25 called NUA sub-address.

To derive this subaddress there are 2 methods:

1) the last 'n' digits are analysed.

2) the base X.25 NUA of the line is removed from the called NUA. e.g. if the called X.25 NUA is 23422120017010 and the PSDN subscriber NUA is 234221200170 then the subaddress derived from subtraction is 10.

Obviously, the second method will not work if the incoming NUA varies.

ISDN Features

ISDN, like X.25, has a core set of features which are then enriched with options. In the original OFTP X.25 specification it was decided that the Q-bit and D-bit options were not common to all networks or applications, they were therefore positively excluded from the specification.

It is proposed that apart from the core ISDN features necessary to establish a call, no other features be used.

Subaddressing

There are two forms of ISDN subaddressing, overdialled and specific.

The overdial method allows an ISDN number to be artificially extended. A typical case would be where a private exchange has been installed in a larger company. Assume that the base number is 394023 and the computer is on internal extension 1234, then by specifying an ISDN number of 3940231234, direct access may be made to the internal extension.

The problem with this method is that it extends to called number and may, especially for international access, exceed the ISDN numbering limits between countries.

The other method of sub-addressing is where a discrete sub address is placed in a specific field in the ISDN call setup.

The problem with this method, is that it requires the caller to place the sub-address in the ISDN call setup. Not all ISDN implementations will allow this insertion.

In conclusion, subaddressing of any kind should be avoided.

Logical Channel Assignment

An X.25 dataline will have associated with it a number of logical channels.

The number of channels is a part of the agreement between the PTT and the subscriber. The number of channels subscribed to is important; call failure and similar problems will result if the number of logical channels defined at the two remote ends are different.

If a DTE makes a call out, then the highest defined logical channel number will be selected, if the remote DCE does not have the same number of logical channels defined, then an invalid logical channel is being used from the perspective of the recipient DCE and the call will be rejected.

Facilities Negotiation

In the PSPDN environment, it is possible to subscribe to negotiation of window size and packet size. Although this negotiation requested by the originator's DTE may be propagated to the remote DTE at the discretion of the originator's DCE, it is a local responsibility between the DTE and DCE pair.

In the ISDN scenario where it is a DTE-DTE type connection, the window size and packet size may be left at the default value and consequently the values may be omitted from the call request. If no values are specified then it is vital that both DTEs have configured themselves to the recommended defaults.

The symptom of a window size mismatch is a hang situation without any informational error codes.

The symptoms of a packet size mismatch could work in some scenarios but would otherwise issue error codes indicating invalid packet sizes.

Window Size

The CCITT X.25 window size has a default value of '2', although subscribers may have other default window sizes, e.g. '7', by virtue of agreement with the PTT.

Window size negotiation can be explicitly requested by specifying the requested window size in the Facilities fields in the Call Request packet.

Packet Size:

The CCITT X.25 packet size has a default value of '128' octets, although subscribers may have other default values, e.g. '1024', agreed with the PTT.

ISDN Call Setup

The initial setup of an ISDN call is initiated with the transmission of a Q.931 SETUP command. Apart from requesting that a call be established, the SETUP command can optionally carry information about the calling party, the called party, routing information, the type of circuit required (e.g. voice or data) and information about the protocols than are requested to be established.

Setup Parameters:

Bearer capability	Information transfer and access attributes
Called Party number	Destination's network address
Called Party subaddress	Destination's complete address
Calling Party number	Source's network address
Low-layer compatibility	Layer 1-3 indication
High-layer compatibility	Layer 4-7 indication

Homologation

Homologation procedures were adopted and vigorously enforced by the PTTs with respect to the quality and conformance of communications equipment connected to the services provided by the PTT s.

In particular, commercial X.25 products had to be tested and approved before they could be connected to the PTTs PSPDN. The advantage of this to the subscriber was that there was very little chance of the approved equipment not working.

With ISDN, similar approval standards are still enforced. So the subscriber has the same confidence in their ISDN equipment. Wrong, the ISDN equipment itself is approved but the X.15 protocol that operates on top of ISDN is now outside of the scope of approval services.

This means that quality of conformance to standards of X.25 over ISDN is subject to the variable quality procedures within the various ISDN equipment manufacturers.

Although it is likely that commercial reputation will place pressure upon the manufacturers with a programming bug to correct such errors, it still requires the subscribers that do not communicate well to put time and effort into finding the party with the error.

So far tests have shown a number of subtle errors, such as timing problems, that have taken many days to find, prove and fix.

Growth

Primary Rate Access:

If a user decides to plan for growth from the beginning, then the Primary Rate Access (PRI) has apparent financial benefits. Such apparent savings are usually lost due to the increased cost of user hardware to support such an interface. The BRI for data usage is very common and cards/adapters are low in cost whereas the PRI cards/adapters are few and far between and consequently highly priced.

Basic Rate Access:

One way to grow with ISDN is to buy multiple BRI lines, increasing slowly in units of 2 x B channels. The PTTs will be able to provide the same subscriber number for all the lines provided in a similar way to the traditional hunting group associated with PSTN type working.

Performance

The obvious benefit of ISDN is speed; unfortunately the majority of computer systems in use today have a finite amount of computing power available. The attachment of multiple active high speed communication lines used in file transfer mode could take a significant amount of CPU resource to the detriment of other users on the system.

Connecting an ISDN line with the default 2 B channels to your computer using an X.21 interface is going to give a consistent 64Kb throughput only if one of the B channels is active at any one time.

If there are two 64Kb channels active and contending for a single 64Kb X.21 interface then effective throughput will be reduced significantly to just over 50 %.

Mainframe issues:

Users with a mainframe front-end are also going to find cost an issue. The scanners that scan the communications interfaces are based upon aggregate throughput. A 64Kb interface takes up a lot of cycles.

Determining 'DTE' or 'DCE' characteristics

The following section is an extract from the ISO/IEC 8208 (International Standards Organization, International

Electrotechnical Commission) (1990-03-15) standard which is an ISO extension of the CCITT X.25 standard.

The restart procedure can be used to determine whether the DTE acts as a DCE or maintains its role as a DTE with respect to the logical channel selection during Virtual Call establishment and resolution of Virtual Call collision.

When prepared to initialise the Packet Layer, the DTE shall initiate the restart procedure (i.e. transmit a RESTART REQUEST packet). The determination is based on the response received from the DXE as outlined below.

a) If the DTE receives a RESTART INDICATION packet with a restarting cause code that is not 'DTE Originated' (i.e., it came from a DCE), then the DTE shall maintain its role as a DTE.

b) If the DTE receives a RESTART INDICATION packet with a restarting cause code of 'DTE Originated' (i.e., it came from another DTE) then the DTE shall confirm the restart an act as a DCE.

c) If the DTE receives a RESTART INDICATION packet with a restarting cause code of 'DTE Originated' (i.e., it came from another DTE) and it does not have an unconfirmed RESTART REQUEST packet outstanding (i.e., a restart collision), then the DTE shall consider this restart procedure completed but shall take no further action except to transmit another RESTART REQUEST packet after some randomly chosen time delay.

d) If the DTE issues a RESTART REQUEST packet that is subsequently confirmed with a RESTART CONFIRMATION packet, then the DTE shall maintain its role as a DTE.

IANA Considerations

This document has no actions for IANA.

Acknowledgements

This document draws extensively on revision 1.4 of the ODETTE File Transfer Specification [OFTP].

Many people have contributed to the development of this protocol and their work is hereby acknowledged.

Informative References

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

[OFTP] Organisation for Data Exchange by Tele Transmission in Europe, Odette File Transfer Protocol, Revision 1.4, April 2000

[FTP] Postel, J., Reynolds, J., "File Transfer Protocol", RFC 959, October 1985,

[RFC-739] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981

[RIME] Coleridge, Samuel Taylor "The Rime of the Ancient Mariner", 1798

[X.509] Internet Society, "Internet X.509 Public Key Infrastructure, Certificate and CRL Profile", RFC 2459, January 1999

[RFC 3850] Internet Society, "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.1 Certificate Handling", RFC 3850, July 2004

Normative References

[CMS Compression] Gutmann, P., "Compressed Data Content Type for Cryptographic Message Syntax (CMS)", RFC 3274, June 2002

[CMS] Housley, R., "Cryptographic Message Syntax (CMS)", RFC 3852, July 2004

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

[PKCS #1] RSA Laboratories East, "PKCS #1: RSA Encryption Version 1.5", RFC 2313, March 1998

[TLS] Internet Society, "The TLS Protocol, Version 1.0", RFC 2246, January 1999

[UTF-8] Yergeau, F., "UTF-8, A Transformation Format of ISO 10646", RFC 3629, November 2003

[ZLIB] Deutsch, P., "ZLIB Compressed Data Format Specification version 3.3", RFC 1950, May 1996

ODETTE Address

The ODETTE File Transfer Protocol is a product of the Technology Committee of Odette International. The Technology Committee can be contacted via the ODETTE Central Office:

ODETTE INTERNATIONAL Limited
Forbes House
Halkin Street

London
SW1X 7DS
United Kingdom

Phone: +44 (0)171 344 9227
Fax: +44 (0)171 235 7112
EMail info@odette.org
Web   www.odette.org

Author's Address

The author can be contacted at:

Ieuan Friend
Data Interchange Plc
Rhys House
The Minerva Business Park
Lynchwood
Peterborough
PE2 6FT
United Kingdom

Phone: +44 (0)1733 371 311
EMail: ieuan.friend@dip.co.uk

IPR Disclosure

Copyright