EUROPEAN TELECOMMUNICATION STANDARD

ETS 300 075

November 1990

Source: ETSI TC/STC TE1

Reference: T/TE 06-04(G)

ICS: 33.020, 33.040.40

Key words: Videotex

Terminal equipment (TE); Videotex processable data

ETSI

European Telecommunications Standards Institute

ETSI Secretariat

Postal address: F-06921 Sophia Antipolis CEDEX - FRANCE
Office address: 650 Route des Lucioles - Sophia Antipolis - Valbonne - FRANCE
X.400: c=fr, a=atlas, p=etsi, s=secretariat - **Internet:** secretariat@etsi.fr

Tel.: +33 92 94 42 00 - Fax: +33 93 65 47 16

0 Foreword

This European Telecommunication Standard (ETS) was produced by the Terminal Equipment (TE) Technical Committee of the European Telecommunications Standards Institute (ETSI) and adopted in September 1990.

This ETS describes the videotex processable data enabling symmetrical file transfer and includes, as a subset, a basic kernal which provides only for file transfer from host to terminal.

This standard is one of an integrated package of 5 ETSs covering various aspects of videotex which comprises:

ETS 300 072 Terminal Equipment (TE);
Videotex presentation layer protocol
Videotex presentation layer data syntax

ETS 300 073 Videotex presentation layer data syntax
Geometric Display
(CEPT Recommendation T/TE 06-02, Edinburgh 1988)

ETS 300 074 Videotex presentation layer data syntax transparent data
(CEPT Recommendation T/TE 06-03, Edinburgh 1988)

ETS 300 076 Terminal Equipment (TE);
Videotex
Terminal Facility Identifier

This standard and its companion ETSs are based on previous CEPT Recommendations and two of them (ETSs 300 073 to 300 074) are CEPT Recommendations now endorsed as ETSs without modification.

Annex A to this ETS, which gives a presently used processable data protocol for file transfer from host to terminal is informative.

For the purposes of this ETS, the reference made in the text to T/TE 06-01 should be read as Final ETS 300 072.

VIDEOTEX PROCESSABLE DATA

Table of Contents

PART1: INTRODUCTION

1 FOREWORD

This standard describes the videotex processable data enabling symmetrical file transfer. This standard includes as a subset a basic kernel which provides only for file transfer from host to terminal.

Part 1, Part 2 and all the notes contained in these two parts form an integral part of this document.

Annex A, which gives a presently used processable data protocol for file transfer from host to terminal, is not an integral part of this standard.

This standard has been prepared by the ETSI STC TE-1 VT group.

2 SCOPE

The videotex processable data facility specified in this document is intended to be used for data file transfer. These files may contain computer software or other file types.

This protocol may be used to download file from the host to the terminal. It may also be used for transfering files between two end systems in both directions, all the operations being then performed under the control of one or the other system depending on a preliminary negociation.

It has been defined to work in a videotex environment but it may also work without specific application environment.

3 RELATED STANDARDS.

CCITT T-101 Rev 1988: International Interworking For Videotex Services.

ETS T/TE 06-01: Videotex presentation layer protocol,
Videotex presentation layer data syntax.

PART 2: SERVICE, APPLICATIONS, PROTOCOLS, CODING

0. GENERAL INTRODUCTION

The Videotex processable data facility specified in this document provides particularly for the transfer of files of data; these files may contain computer software, but other file types are not precluded. This facility also provides for data to be reliably transferred to devices associated with a videotex terminal under control of the received data. In addition to transparent transfer, specific standardized provisions are made for passing data to an associated printer, but the protocol is not limited to this device, and other devices may be standardized later.

Two service classes are currently defined in this document:

- A "basic kernel" provides for file transfer from a host to a terminal, all the operations being controlled by the host. The access to a file or to a specific set of files is carried out by performing a videotex dialogue which takes place outside the downloading phase and which is not part of the current specification. Moreover this basic kernel provides for low level recovery mechanisms.

- An enhanced service, called "symmetrical service", provides for transferring files between two systems in both directions, all the operations being able to be performed under the control of one or the other systems according to a preliminary negotiation. The access to a file or to a specific set of files is carried out with a dialogue phase which can be completely automatized and which is part of the dowloading protocol. In this service class enhanced facilities are available: recovery, user identification, window mechanism.

These two service classes allow to support a wide range of applications which require file transfer. Two applications are currently defined in this document: a telesoftware application and a file transfer application intended for printing. These applications are specified as rules for the use of the T-service and as structure and coding of the exchanged files.

Section 1 of this document describes the functions which are offered to an application using the processable data facility. These functions are described in terms of service elements.

In order to allow for the transfer of files in already existing videotex systems as well as allowing more advanced facilities for future systems, the transfer is described as consisting of two layers.

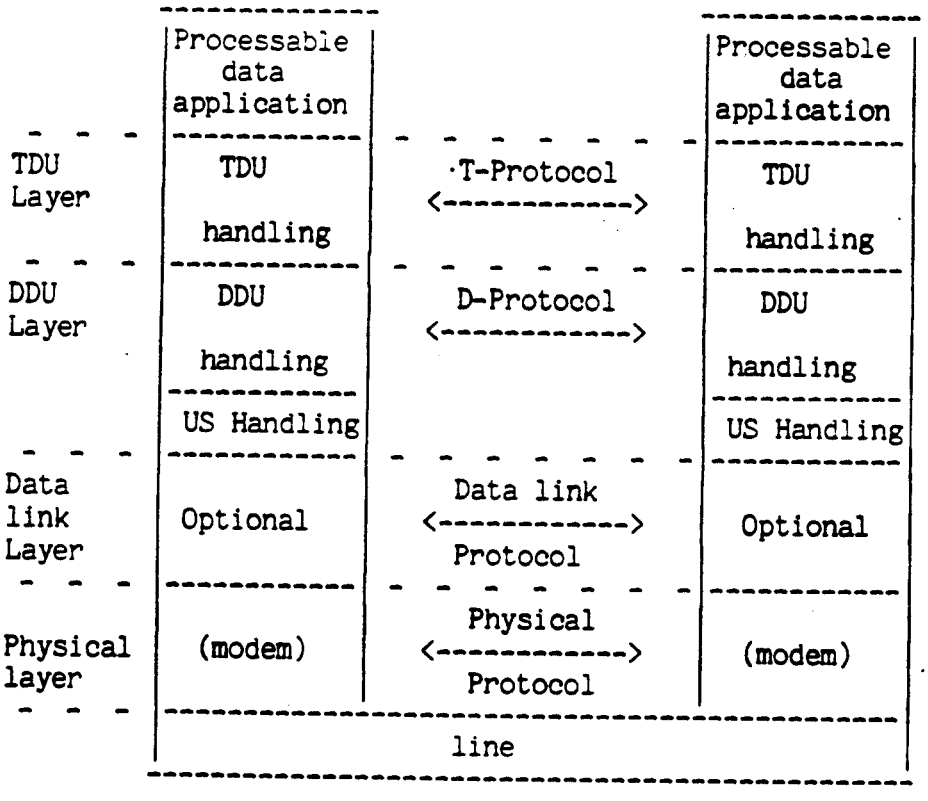
Processable data, including telesoftware files, data for printing, file parameters and control data related to the dowloading procedure are transmitted by means of Telesoftware Data Units (TDU's). These TDU's are exchanged between cooperating entities according to the

T-protocol. This protocol, as well as the specific rules for telesoftware and printer device applications are specified in section 3 of this document.

In addition Dialogue Data Units (DDU's) are used for adaptation to the different Videotex systems. Optional 8-bit transparency capabilities and optional error detection and recovery facilities are provided. Handling of these DDU's are described in section 5 of this document.

Figure 1 describes the theoretical model for handling processable data.

FIGURE 1 - THEORETICAL MODEL OF PROCESSABLE DATA HANDLING



1 SERVICE DEFINITION

1.1 SCOPE AND FIELD OF APPLICATION

This section defines in an abstract way the externally visible service provided by the TDU layer (T-service) in terms of:

a) the primitive actions and events of the service,
b) the parameter data associated with each primitive action and event,
c) the relationship between, and the valid sequence of these actions and events.

This section also describes the processable data applications which make use of the above mentioned service.

1.2 MODEL OF THE T-SERVICE

1.2.1 Service references

This document is based on the concepts which were developed in CCITT and ISO for the description of the OSI Reference model (CCITT Rec. X.200, ISO 7498).

The conventions used to describe this service are based on CCITT Recommendation X.210 (OSI layer service definition conventions).

1.2.2 Service definitions

1.2.2.1 General terms

This section makes use of the following terms as they are defined in Recommendation X.210:

a) service user
b) service provider
c) primitive
d) request
e) indication
f) response
g) confirmation
h) confirmed, non confirmed, provider initiated service

The following definitions also apply:

Optionally confirmed service: confirmed or non confirmed service according to the user's choice specified in the request primitive.

Protocol phase: period of time during which the exchanges are dedicated to a specific function (connection, disconnection, mass transfer ...).

Regime: set of protocol phases; a regime is a continuous period of time. A Regime is established by using a confirmed or optionally confirmed service and it is orderly terminated by using a confirmed service, it may also be interrupted in an abnormal manner.

A regime is fully defined by specifying the service(s) used to establish it and the service(s) used to terminate it. A regime is used in this description to limit the range of some services which may only be available during a particular regime.

Transfer unit: data transferred by using one mass transfer primitive.

Initiator: the entity which initiates a service request.

Acceptor: the entity which accepts (or refuses) a service indication.

1.2.2.2 Regimes

Three regimes are defined: the Association regime, the Access regime and the Transfer regime. In the basic kernel only the Association and the Transfer regimes may be established.

These regimes are defined in section 1.2.3.

In the following, the functions of the regimes and their relationship with each other are specified.

An Association regime determines a period during which two applications remain associated.

An Access regime is used to allow functions to be negotiated and it determines a period during which these functions are available. The Access regime is never established in "basic kernel".

A Transfer regime determines a period during which a mass transfer is performed.

The mass transfer phase is related to the data transfer itself and takes place during a Transfer regime.

An Access regime is established within an Association regime, provided no other Access regime is already established. A Transfer regime is established within an Access regime (symmetrical service) or within an Association regime (basic kernel), provided no other Transfer regime is already established.

Figures 2 and 3 show examples of establishing regimes in the basic kernel and in the symmetrical service.

FIGURE 2- EXAMPLE OF REGIMES ESTABLISHMENT IN THE BASIC KERNEL

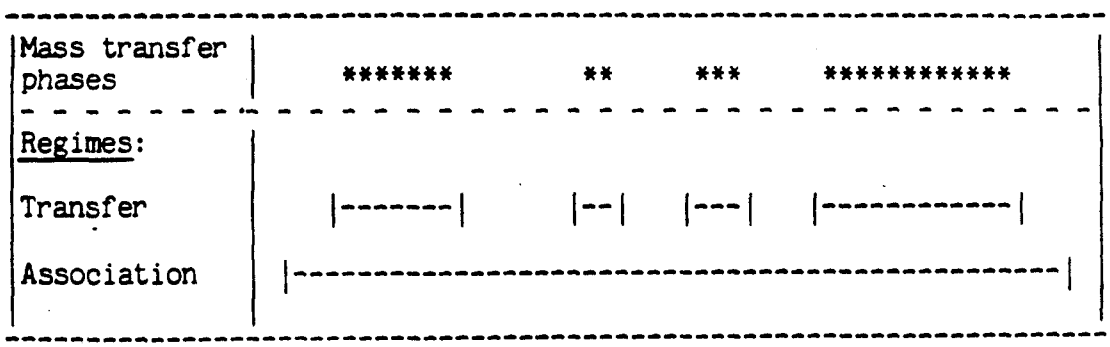
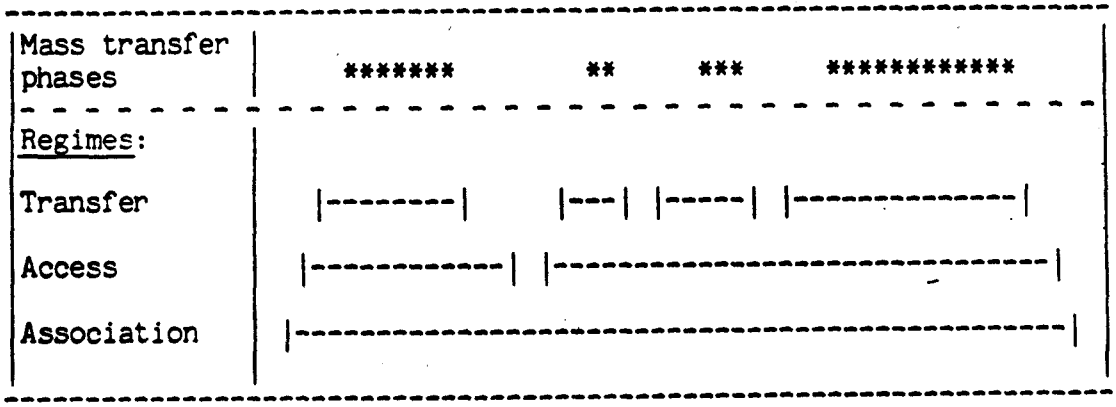


FIGURE 3- EXAMPLES OF REGIMES ESTABLISHMENT IN THE SYMMETRICAL SERVICE



1.2.2.3 Roles

At a given time each entity is assigned a unique role. Within a given regime, a role determines the set of services for which the entity may be initiator or acceptor.

The following roles are defined:

- The **Master** is the entity which controls the dialogue.

- The **Slave** is the entity which performs the operations requested by the Master.

- The **Sender** is the entity which sends the data during a mass transfer.

- The **Receiver** is the entity which receives the data during a mass transfer.

During a whole Transfer regime a given entity keeps the same sender or receiver role.

After having established an Association regime the Master role is assigned to the initiating entity of the association establishement service, the Slave role is assigned to the accepting entity of the association establishment service.

At the Access regime establishment (in symmetrical service) the Master and Slave roles may be modified and will then remain unchanged during the whole Access regime. After the end of the Access regime the Master role is assigned to the initiating entity of the Access regime release service, the Slave role is assigned to the accepting entity of the Access regime release service.

At the Transfer regime establishment the Sender and Receiver roles are assigned according to the mass transfer direction. The mass transfer direction is determined by the service which has been used to establish the Transfer regime. At the end of the Transfer regime, each entity takes its own Master or Slave role which was assigned before the Transfer regime establishment.

In the service class "basic kernel", the mass transfer is always performed in the same direction, therefore the Sender role is always assigned to the Master, the Receiver role is always assigned to the Slave.

1.2.2.4 Local concepts

- The **Server** is the system which contains a database (which stores and retrieves information without processing it). In the basic kernel the **Server** may be called the **Host**.

- The **Executor** is the system which processes the downloaded files.

A given system may contain both a Server application and an Executor application. The Server and Executor concepts are local concepts and are not related with file transmission, however this may impact implementation subsets definition.

In the service class "basic kernel", during a whole association, the Server will play the Master and Sender roles, while the other entity is the Executor and will play the Slave and Receiver roles.

1.2.3 Service elements

1.2.3.1 General organization

Table 1 gives the list of the service elements:

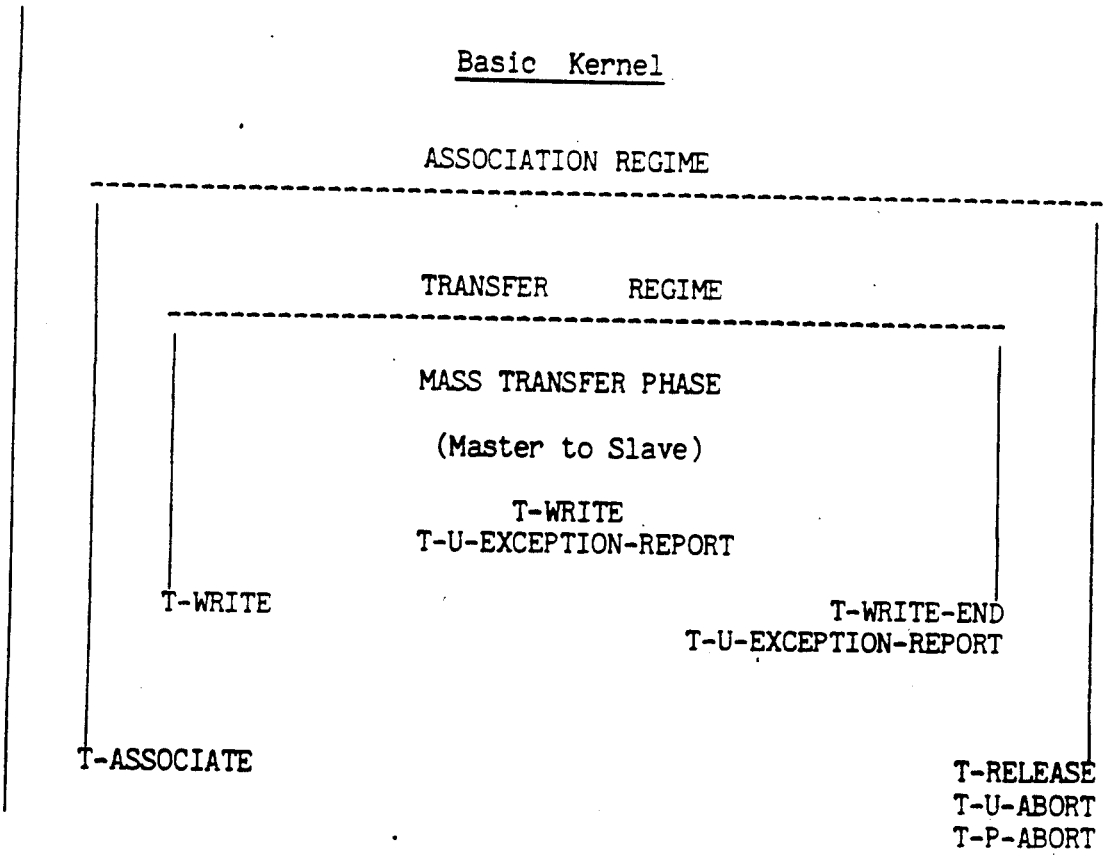
TABLE 1 - LIST OF THE SERVICE PRIMITIVES

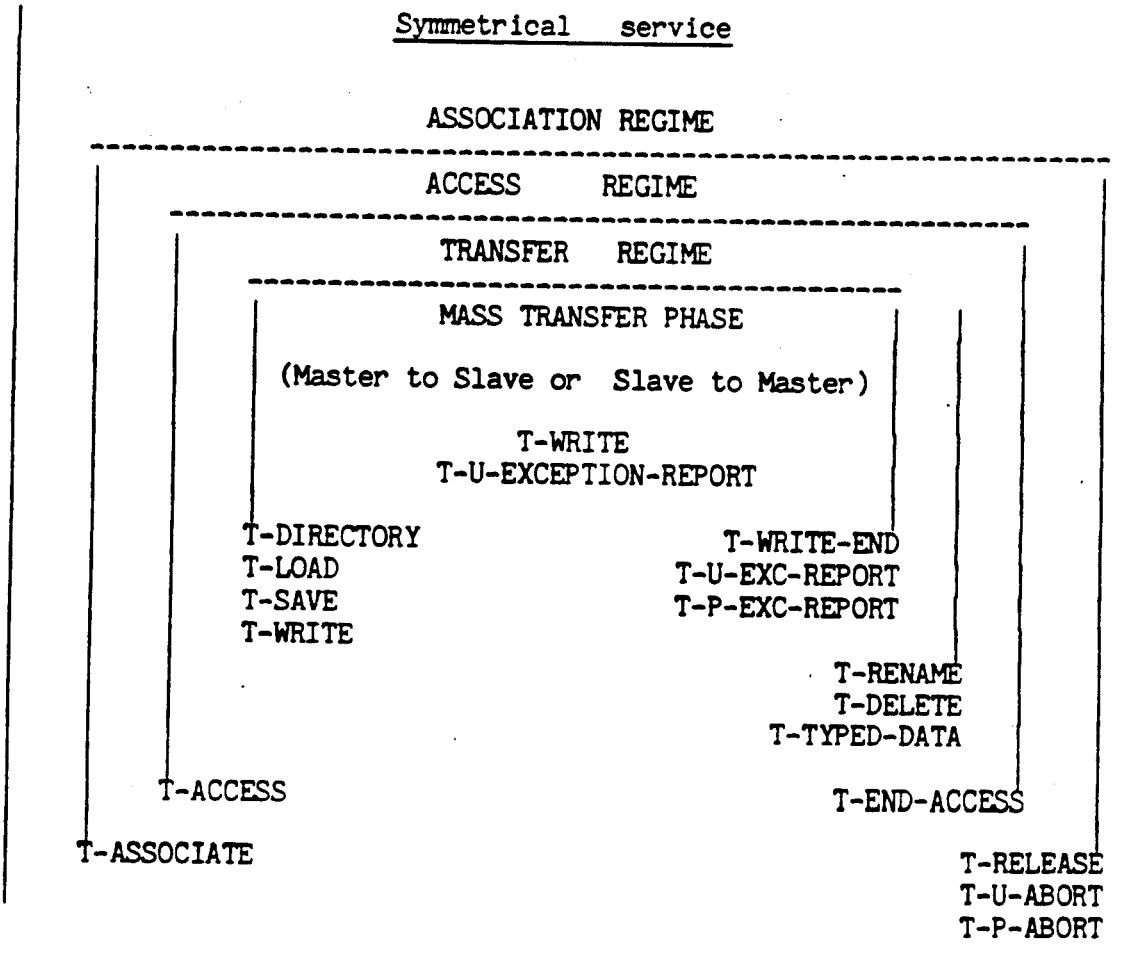
Service		initiated by	Function	S	B
T-ASSOCIATE	OC	both	Association establishment	M	M
T-RELEASE	C	both(1)	Association release	M	M
T-U-ABORT	NC	both(2)	Association user abort	M	M
T-P-ABORT	P	both	Association provider abort	M	-
T-ACCESS	C	Master	Access regime establishment	M	-
T-END-ACCESS	C	both	End of Access regime	M	-
T-DIRECTORY	C	Master	File Directory request	O	-
T-LOAD	C	Master	Init. Slave to Master mass transfer	O	-
T-SAVE	C	Master	Init. Master to Slave mass transfer	O	-
T-RENAME	C	Master	Rename file	O	-
T-DELETE	C	Master	Delete file	O	-
T-TYPED-DATA	NC	both	Typed data transfer	O	-
T-WRITE	OC	Sender	Data transfer	M	M
T-WRITE-END	C	Sender	End of data transfer	M	M
T-U-EXCEPT.	NC	both(3)	User exception report	M	M
T-P-EXCEPT.	P	both	Provider exception report	M	-

C : Confirmed service
OC : Optionally confirmed service
NC : Non confirmed service
P : Provider initiated service
B : Basic kernel
S : Symmetrical service
O : Optional
M : Mandatory
(1): May only be sent by the Master in the basic kernel
(2): May only be sent by the Slave in the basic kernel
(3): May only be sent by the Receiver in the basic kernel
- : Irrelevant

Figure 4 gives the relationship between services and regimes, indicating which services are used to establish and terminate a regime and which services are available in a given regime. The definition of the regimes is given in the following sections.

FIGURE 4 - REGIMES





Note 1.1: T-U-ABORT and T-P-ABORT may be used at any time in every regime, they terminate all the currently established regimes.

1.2.3.2 Association regime

The Association regime is established by using the association establishment service. It may be terminated by using the association release service or the association abort services.

In the idle state the Master may invoke the association abort or release services, the Access (symmetrical service) or Transfer (basic kernel) regime establishment services.

In the idle state the Slave may invoke the association abort or release services.

1.2.3.3 Access regime

The Access regime is established (in symmetrical service) by using the Access regime establishment service. It may be terminated by using the end of Access regime service or the association abort service (in this latter case the association is also terminated).

When the Access regime is established, the Master and the Slave may invoke the end of Access regime service, the association abort service or the exception report services. The Master may invoke the data transfer, load, save, rename, delete, file directory services, the Master and the Slave may also invoke the typed data transfer service provided that the use of those services had been negotiated during the Access regime establishment.

1.2.3.4 Transfer regime

The Transfer regime is used to transmit a large amount of information (e.g. files) from the Sender to the Receiver. The mass transfer phase, which consists in performing the data transfer and end of data transfer services, takes place during the transfer regime.

In the symmetrical service class, a Transfer regime may be established within the Access regime by using the load, save or file directory services. When the Access regime is established, the Transfer regime may also be established implicitly by the Master by starting the mass transfer phase. When a Transfer regime consists only in a mass transfer phase (i.e. it is directly established by issuing a data transfer service primitive) the transfer is called Basic Transfer Mode. In the symmetrical service class, the use of the Basic transfer mode is negotiated at the Access regime establishment and is exclusive of the use of file directory, load or save services.

In the basic kernel, a Transfer regime is reduced to the mass transfer phase (Basic transfer mode). Therefore the Transfer regime is established when the Association regime is established by starting the mass transfer phase.

A Transfer regime may be terminated by using the end of data transfer service, the exception report service or the association abort services.

During the Transfer regime, the Sender may invoke the data transfer service, the end of data transfer service, the exception report services or the association abort services. The Receiver may invoke the exception report services and the association abort services. In the symmetrical service only the Receiver may invoke the user exception report service.

1.2.3.5 Restrictions on the use of services

Sections 1.2.3.2, 1.2.3.3, 1.2.3.4 specify which services may be used in each regime.

Moreover, between sending a confirmed service request and receiving the corresponding confirmation (at the service initiator side) or between receiving a confirmed service indication and sending the corresponding response (at the service acceptor side), no other service may be initiated except the exception report, or association abort services. This restriction is not applicable to the data transfer services when the window size is greater than 1 and in the conditions described in section 1.2.4.3.

1.2.4 Concepts related to mass transfer

In order to facilitate the transfer of a large amount of information various mechanisms are provided.

The main notion is the recovery point. The recovery points are located at the beginning of the mass transfer phase and at each confirmed data transfer primitive.

In the symmetrical service, the size of transfer units (conveyed within a data transfer primitive) is negotiated during the Access regime establishment.

1.2.4.1 Recovery during a mass transfer

Several facilities are provided to recover during a mass transfer phase:

- The Receiver may request to restart the transmission at the beginning of the mass transfer phase. This facility is always available in the basic kernel and it is negotiated during the Access regime establishment in the symmetrical service.

- During the mass transfer, the transmission may be resumed from the last confirmed data transfer primitive, this may be done by sending a negative response to a data transfer primitive. The transmission is resumed from the data which immediately followed the last data for which a positive confirmation had been received by the Sender or, if not, at the beginning of the transfer.

1.2.4.2 Recovery outside a mass transfer

The recovery outside a mass transfer phase is carried out when the mass transfer phase has been interrupted. This recovery may take place during the same association or during another association than the original transfer. This mechanism is only available in the symmetrical service when using Load or Save services and if it is negotiated during the Access regime establishment.

When the recovery outside a mass transfer has been negotiated, the Sender must associate a recovery point number to each transfer unit and each data transfer primitive must be confirmed.

Note 1.2: The service does not ensure that the information for which the transmission is resumed are consistent with the previously transmitted information. It is up to the service user to provide means for a correct recovery (e.g. storage of the interrupted transfer context, file version number ...).

The recovery request is performed by indicating a recovery point number in a parameter of the T-SAVE request or T-LOAD request primitives. The transmission is resumed starting with the data which immediately followed the indicated recovery point in the original transfer. The indicated recovery point corresponds to the last data transfer primitive for which a positive confirmation had been received or sent.

Note 1.3: After a transfer interruption, it may happen that the recovery point number, from the Sender point of view, will be lower than the recovery point number from the Receiver point of view. It is up to the Receiver to verify that no data duplication occurs in case of recovery.

1.2.4.3 Anticipation window

The anticipation mechanism gives the ability to send several data transfer primitives with explicit confirmation requested, without waiting for having received the confirmation of the previous ones. The aim of the window mechanism is to improve the transmission efficiency by avoiding idle periods due to the waiting of confirmations.

The anticipation window is the number of recovery point (i.e. the number of data transfer primitives with explicit confirmation requested) which the Sender may send without having received any confirmation. When there is no anticipation the window size is equal to 1.

When the anticipation window size is greater than 1, the Sender must associate a recovery point number to each transfer unit and each data transfer primitive must be confirmed.

In the basic kernel the window size is always equal to 1.

In symmetrical service, the window size is negotiated during the Access regime establishment. Each entity indicates the window size it may accept when the entity plays the role of Receiver. However the Sender may use, for sending, a window of a lower value than the value indicated by the Receiver (i.e. the Sender is not bound to fill in the sending window). The Receiver must confirm the recovery points "as soon as possible" and should not wait until the window maximum value is reached to confirm them (this value may never be reached if the Sender uses a lower window size for sending).

When the value of the maximum window size is reached the Sender is no longer permitted to send data until it receives an explicit confirmation to a previous recovery point. The recovery points are set by incrementing the recovery point number by one for each new recovery point and they are explicitely confirmed (one by one) in the increasing order of their reception.

1.3 ASSOCIATION REGIME CONTROL

If the association establishment service requires confirmation, the association regime is established as soon as the acceptor has sent a T-ASSOCIATE Response (positive) and, at the initiator's end, a T-ASSOCIATE Confirmation (positive) has been received.

If the association establishment does not require confirmation, the association regime is established as soon as the initiator has sent a T-ASSOCIATE Request and, at the acceptor's end, a T-ASSOCIATE Indication has been received.

The association regime is terminated upon transmission of T-RELEASE Response or T-U-ABORT Request and upon reception of T-RELEASE Confirmation, T-U-ABORT Indication or T-P-ABORT Indication.

1.3.1 **Association establishement**

1.3.1.1 Function

T-ASSOCIATE is used by a service user to associate with a specified processable data application (identified by the "application name" parameter).

The T-ASSOCIATE Request primitive is used to establish an Association regime, this primitive may not be used when the association is being established or is already established.

In the basic kernel, a T-ASSOCIATE request may only be initiated by the host.

T-ASSOCIATE is an optionally confirmed service, the parameter "explicit confirmation" indicates whether or not an explicit confirmation is requested.

1.3.1.2 Parameters

The different primitives and parameters required by the association establishment service are described in the following table.

Parameter	T-ASSOCIATE Request	T-ASSOCIATE Indication	T-ASSOCIATE Response	T-ASSOCIATE Confirm.	S
Service class	Mandatory	Mandatory(=)	Mandatory	Mandat.(=)	
Called address	Optional	Optional(=)	Optional	Optional(=)	S
Calling addr.	Optional	Optional(=)			S
Appl. name	Mandatory	Mandatory(=)			
Explicit conf.	Mandatory	Mandatory(=)			
Timeouts	Optional	Optional(=)	Optional	Optional(=)	
Request ident.	Optional	Optional(=)			S
Identification	Optional	Optional(=)	Optional	Optional(=)	S
User data	Optional	Optional(=)	Optional	Optional(=)	
Result			Mandatory	Mandatory	

(=): The value of the parameter is identical to the value of the corresponding parameter in the preceding request or response primitive.
S : Symmetrical service only.

1.3.1.2.1 Service class

This parameter is used to indicate the service class in use. It may take the following values:

a) "basic kernel"
b) "symmetrical service"

In the Request/Indication it indicates the service class(es) proposed.

In the Response/Confirmation it announces the service class selected if the association is accepted.

1.3.1.2.2 Called address

This parameter indicates the address of the entity who has sent a response or a confirmation. this address may differ from the parameter provided in the request or the indication (this can be the case after rerouting).

This parameter is not used if the service class parameter indicates only the value "basic kernel".

1.3.1.2.3 Calling address

This parameter provides the address of the entity who originated the association.

This parameter. is not used if the service class parameter indicates only the value "basic kernel".

1.3.1.2.4 Application name

The parameter "application name" announces the use of a specified processable data application. This parameter is a variable length string.

The following standardized names are defined:

!A : Any application (including telesoftware and printer).
!T : Telesoftware .
!P : Printer device.
Names starting with ! are reserved for standardized applications.

1.3.1.2.5 Explicit confirmation

This parameter is used in the Basic Kernel to indicate whether or not explicit confirmation is requested.

If the service class parameter indicates "Symmetrical Service", this parameter must be set to indicate that confirmation is requested.

1.3.1.2.6 Timeouts

In the basic kernel, this parameter indicates the master's maximum response time and contains the maximum value of the delay between a response primitive issued by the slave and the next indication primitive. This delay is controlled by the service provider. Absence of this parameter means that this delay must not be controlled by the service provider.

In the symmetrical service, this parameter specifies the maximum response time required to answer a request. Each entity is responsible for defining their own response time. This response time is controlled by the service provider at the other end.

Absence of this parameter means in both cases, that the response time should not be checked by the service provider.

1.3.1.2.7 Request identification

This parameter enables the initiator to require identification in the Response/Confirmation. It may take the two following values:

a) identification required,
b) identification not required.

This parameter is not used if the service class parameter indicates only the value "basic kernel".

1.3.1.2.8 Identification

This parameter is a variable length string, of no more than 254 bytes. It provides identification data on the user of this service.

This parameter is not used if the service class parameter indicates only the value "basic kernel".

1.3.1.2.9 User data

This parameter is used to convey string type information of no more than 254 bytes.

In the basic kernel this parameter cannot be used in the T-ASSOCIATE response and T-ASSOCIATE confirmation.

1.3.1.2.10 Result

This parameter indicates whether the association is accepted or rejected.

If the selected service class is the symmetrical service, possible values of this parameter are the following:

a) association request accepted.

b) association request rejected. Reason not specified.

c) Reject from the service provider, reason:

called address incorrect
calling address incorrect
service class refused

d) Reject from the service user, reason:

called address incorrect
calling address incorrect
service class refused
application's name unknown
wrong identification
erroneous user data
other reason.

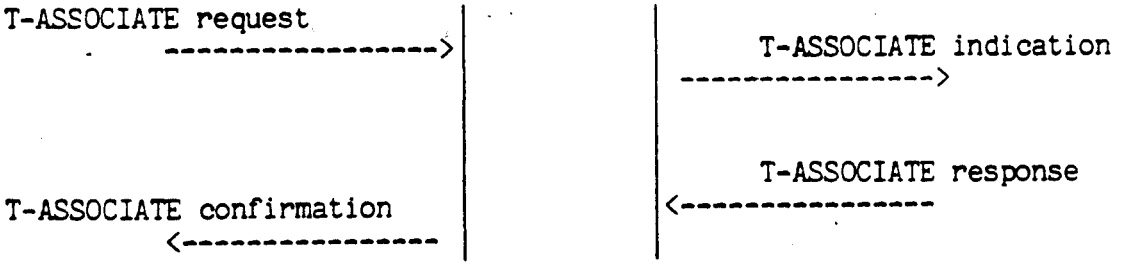
If the selected service class is the basic kernel, the value of this parameter can be either:

a) association request accepted

b) association request rejected.

1.3.1.3 Association establishment operation

In the case when an explicit confirmation is requested by the parameter "explicit confirmation", the corresponding response primitive should be sent by the receiver before any other protocol exchange.



If explicit confirmation is not requested and if the application specified in a T-ASSOCIATE is not available at the acceptor's side or in the case of any user error, then the acceptor should use the T-U-ABORT service.

1.3.2 Association release

1.3.2.1 Function

An Association regime may be terminated by the exchange of T-RELEASE primitives. This service may be used only once the Association regime is established and provided no other regime is established.

The T-RELEASE service is thus used to request the orderly termination of the processable data application.

If the selected service class is the basic kernel only the Master may issue a T-RELEASE request.

1.3.2.2 Parameters

The different primitives and parameters required by the association release service are described in the following table.

Parameter	T-RELEASE Request	T-RELEASE Indication	T-RELEASE Response	T-RELEASE Confirm.
User data	Optional	Optional(=)	Optional	Optional(=)
Result			Mandatory	Mandatory

(=): The value of the parameter is identical to the value of the corresponding parameter in the preceding request or response primitive.

1.3.2.2.1 User data

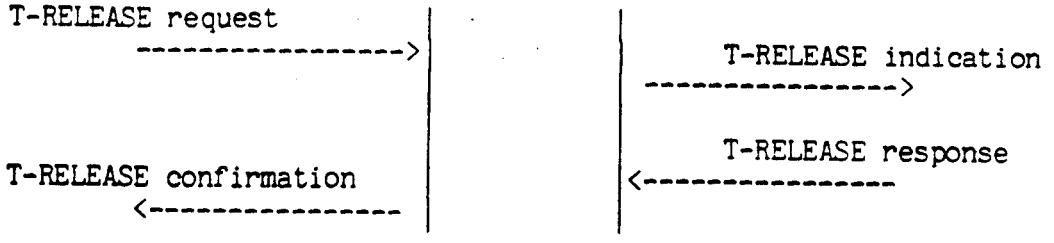
This parameter is used to convey string type information of no more than 254 bytes.

In the basic kernel, this parameter cannot be used in the T-RELEASE response and T-RELEASE confirmation.

1.3.2.2.2 Result

This parameter always indicates that the T-Release service has been accepted.

1.3.2.3 Association release operation



1.3.3 Association abort

1.3.3.1 Function

This service is performed by the primitives T-U-ABORT Request when it is user initiated, by T-P-ABORT when it is provider initiated.

In the basic kernel a T-U-ABORT request may only be initiated by the slave. T-P-ABORT is not used in the basic kernel.

A T-U-ABORT request may be sent by the user of the T-Service at any time after reception of a T-ASSOCIATE Indication or after transmission of a T-ASSOCIATE Request. The provider of the T-Service may issue a T-P-ABORT Indication at any time after reception of a T-ASSOCIATE Request or after transmission of T-ASSOCIATE Indication.

It is recommended to avoid the use of the T-U-ABORT service to reject a T-ASSOCIATE for which explicit confirmation was requested.

These primitives terminate the association regime abnormally. Data may be lost.

1.3.3.2 Parameters

The different primitives and parameters required by the abort service are described in the following table.

Parameter	T-U-ABORT request	T-U-ABORT indication
Reason		Mandatory

Reason (user):

reason not specified,
incorrect identification,
incorrect role/function,
other reason.

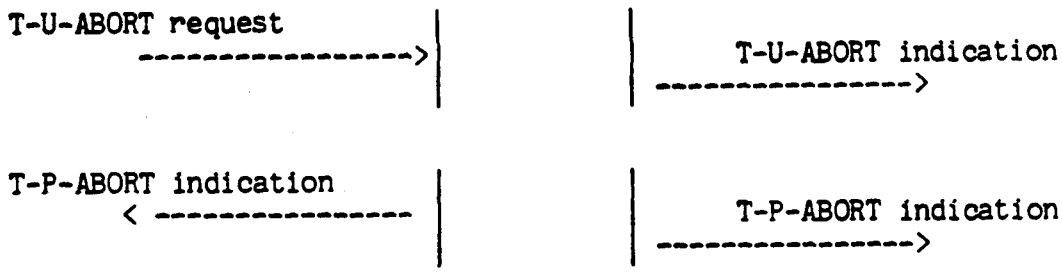
If the service class is the basic kernel this parameter is always equal to "reason not specified".

Parameter	T-P-ABORT indication
Reason	Mandatory

Reason (provider):

reason not specified,
repeated negative acknowledgements/repeated errors,
delay expired,
unknown message,
protocol conflict,
lower layer error,
syntax error/parameter's absence,
other reason.

1.3.3.3 Association abort operation



1.4 ACCESS REGIME CONTROL.

The Access regime is established upon transmission of the primitive T-ACCESS Response (positive) on the acceptor's side and upon reception of the primitive T-ACCESS Confirmation (positive) on the initiator's side.

The Access regime is terminated upon transmission of the primitive T-END-ACCESS Response or T-U-ABORT Request (on the acceptor's side) and upon reception of the primitive T-END-ACCESS Confirmation, T-U-ABORT Indication or T-P-ABORT Indication (on the initiator's side).

1.4.1 Access establishment.

1.4.1.1 Function.

The T-ACCESS service enables the establishment of the Access regime provided the service class selected is the symmetrical service.
The establishment of this regime is initiated by the primitive T-ACCESS Request, this primitive may only be issued by the Master in the Association regime and provided no other Access regime is established.

The establishment of the Access regime can be rejected by the acceptor by use of the primitive T-ACCESS Response. In such case, after completing the T-ACCESS service, both the entities are idle in the Association regime.

In case the initiator of the T-ACCESS primitive is unsatisfied with the answer, it may issue a T-END-ACCESS Request primitive thus terminating the Access regime and supplies the reason of the termination.

1.4.1.2 Parameters.

The different primitives and parameters required by the Access service are described in the following table.

Parameters	T-ACCESS Request	T-ACCESS Indication	T-ACCESS Response	T-ACCESS Confirm.
Role	Mandatory	Mandatory(=)		
Functions	Mandatory	Mandatory(=)	Mandatory	Mandatory(=)
Transfer unit size	Mandatory	Mandatory(=)	Mandatory	Mandatory(=)
Anticipation window	Mandatory	Mandatory(=)	Mandatory	Mandatory(=)
Recovery	Mandatory	Mandatory(=)	Mandatory	Mandatory(=)
Transfer Mode	Mandatory	Mandatory(=)	Mandatory	Mandatory(=)
User data	Optional	Optional(=)	Optional	Optional(=)
Result			Mandatory	Mandatory(=)

(=): The value of the parameter is identical to the value of the corresponding parameter in the preceding request or response primitive.

1.4.1.2.1 Role

This parameter can take either of the two following values:

a) Master

b) Slave

This parameter indicates the role adopted by the initiator of the T-ACCESS Request primitive. This role will replace the role previously assigned as soon as the Access regime is established.

1.4.1.2.2 Function.

This parameter indicates whether or not the user handles the following Indication primitives:

- Slave: T-DIRECTORY, T-LOAD, T-SAVE, T-RENAME, T-DELETE, T-TYPED DATA, T-U-EXCEPTION (Read Restart).

- Master: T-TYPED DATA, T-U-EXCEPTION (Read Restart).

If the Basic Transfer mode is selected (see 1.4.1.2.6), then the T-DIRECTORY, T-LOAD and T-SAVE services must not be used.

1.4.1.2.3 Size of transfer units.

This parameter indicates the maximum size of the application data contained in a T-WRITE or T-WRITE-END Request primitive.

The following values are permitted:

512, 1024, 2048, 4096, 8192,16384, 32768, 65536 bytes.

In the basic kernel this parameter is not used and the value is 1024 bytes.

1.4.1.2.4 Anticipation window.

This parameter indicates the maximum number of consecutive T-WRITE or T-WRITE-END (confirmation required) primitives the user is allowed to receive before issuing an answer.
It may take a value ranging from 1 to 8.

This parameter is not used in the basic kernel, the value is 1.

1.4.1.2.5 Recovery.

This parameter indicates whether or not the slave is able to handle the recovery mechanism upon transmission and reception of files.

1.4.1.2.6 Transfer mode

This parameter may take one of the two following values:

a) Basic Transfer Mode supported (Slave) or Basic Transfer Mode required (Master).

b) Basic Transfer Mode not supported (Slave) or Basic Transfer Mode not required (Master).

If the Master indicates that it requires the use of the Basic Transfer Mode and the Slave indicates that it supports this mode, then the Basic Transfer Mode is selected which means that only this mode may be used to transfer files and that the Load, Save or Directory functions must not be used during the Access regime whatever the Slave indicated in the Function parameter. Otherwise the Basic Transfer Mode must not be used (only the other functions indicated in the Function parameter may be used).

1.4.1.2.7 User data

This parameter is used to convey string type information of no more than 254 bytes.

1.4.1.2.8 Result

This parameter indicates whether or not the service has been accepted or rejected.

This parameter may take one of the following values:

a) Access request accepted.

b) Access request rejected. Reason not specified.

c) Refusal from the service provider, reason:

role refused.

d) Refusal from the service user, reason:

role refused,
insufficient primitives handled,
erroneous user data,
other reason.

1.4.2 End of Access service.

1.4.2.1 Function.

The Access regime may be terminated using a T-END-ACCESS primitive. The T-END-ACCESS Request primitive may only be issued within the idle state of the Access regime.

It may be initiated by both the Master and the Slave.

1.4.2.2 Parameters.

The different primitives and parameters required by the End Access service are described in the following table.

Parameters	T-END-ACCESS Request	T-END-ACCESS Indication	T-END-ACCESS Response	T-END-ACCESS Confirm.
User data	Optional	Optional(=)	Optional	Optional(=)
Result	Mandatory	Mandatory(=)		

(=): The value of the parameter is identical to the value of the corresponding parameter in the preceding request or response primitive.

1.4.2.2.1 User data

This parameter is used to convey string type information of no more than 254 bytes.

1.4.2.2.2 Result

This parameter indicates the reason of the Access regime termination.

reason not specified,
insufficient primitives handled,
other reason.

1.4.3 File directory service.

1.4.3.1 Function.

The File directory service enables the Master to request the transfer of one file directory from the Slave to the Master and the establishment of the Transfer regime. This service is not available in the Basic Kernel.

Only the Master may issue a T-DIRECTORY Request, provided the Access regime is established and the use of T-DIRECTORY has been accepted by the Slave at the time the Access regime was established.

The information is transfered from the Slave to the Master after the Slave has answered by T-DIRECTORY Response (accept). A positive response establishes the Transfer regime.

In the Transfer regime the Slave becomes Sender and the Master Receiver.

1.4.3.2 Parameters.

The different primitives and parameters required by the File directory service are described in the following table.

Parameters	T-DIRECTORY Request	T-DIRECTORY Indication	T-DIRECTORY Response	T-DIRECTORY Confirm.
Designation	Mandatory	Mandatory(=)		
User data	Optional	Optional(=)		
Result			Mandatory	Mandatory(=)

(=): The value of the parameter is identical to the value of the corresponding parameter in the preceding request or response primitive.

1.4.3.2.1 Designation.

This parameter enables the Master to designate the directory it requires. It is a byte sequence of no more than 254 bytes.

1.4.3.2.2 User data

This parameter is used to convey string type information of no more than 254 bytes.

1.4.3.2.3 Result

This parameter enables the Slave to indicate acceptance or refusal of the directory request.

This parameter may take one of the following values:

a) acceptance,

b) user refusal, reason:

reason not specified,
erroneous designation,
no answer to the request,
erroneous user data,
other reason.

1.4.4 Load service.

1.4.4.1 Function.

The Load service enables the Master to request the transfer of one file from the Slave to the Master and establishes the Transfer regime.
The Load service is not used if the Basic Transfer Mode is used (the Basic Transfer Mode is used in the Basic Kernel or as an option of the Symmetrical Service).

Only the Master may issue a T-LOAD Request, provided the Access regime is established and the use of T-LOAD has been accepted by the Slave at the time the Access regime was established.

The information is transferred from the Slave to the Master after the Slave has answered by T-LOAD Response (accept). A positive response establishes the Transfer regime.

In the Transfer regime the Slave becomes Sender and the Master Receiver.

1.4.4.2 Parameters.

The different primitives and parameters required by the Load service are described in the following table.

Parameters	T-LOAD Request	T-LOAD Indication	T-LOAD Response	T-LOAD Confirm.
Recovery point	Optional	Optional(=)		
Designation	Mandatory	Mandatory(=)		
User data	Optional	Optional(=)		
Result			Mandatory	Mandatory(=)

(=): The value of the parameter is identical to the value of the corresponding parameter in the preceding request or response primitive.

1.4.4.2.1 Recovery point.

This parameter is used to enable the Master to indicate the recovery point which corresponds to the last data transfer primitive for which a positive confirmation had been sent. The Transfer will be resumed from the data transfer service immediately following the indicated recovery point. The value of the recovery point is in the range 0 to 65535.

This parameter cannot be used if the recovery function has not been accepted at the time the Access regime was established.

1.4.4.2.2 Designation.

This parameter enables the Master to designate the information it requires. It is a byte sequence of no more than 254 bytes.

1.4.4.2.3 User data

This parameter is used to convey string type information of no more than 254 bytes.

1.4.4.2.4 Result

This parameter enables the Slave to indicate acceptance or refusal of the load request.

This parameter may take one of the following values:

a) acceptance,

b) user refusal, reason:

reason not specified,
erroneous designation,
unknown file,
erroneous recovery point,
erroneous user data,
other reason.

1.4.5 Save service.

1.4.5.1 Function.

The Save service enables the Master to request the transfer of one file from the Master to the Slave and establishes the Transfer regime. This service is not available in the Basic Kernel.

Only the Master may issue a T-SAVE Request, provided the Access regime is established and the use of T-SAVE has been accepted by the Slave at the time the Access regime was established.

The information is transferred from the Master to the Slave after the Slave has answered by T-SAVE Response (accept). A positive response establishes the Transfer regime.

In the Transfer regime the Slave becomes Receiver and the Master Sender.

1.4.5.2 Parameters.

The different primitives and parameters required by the Save service are described in the following table.

Parameters	T-SAVE Request	T-SAVE Indication	T-SAVE Response	T-SAVE Confirm.
Recovery point	Optional	Optional(=)		
Designation	Mandatory	Mandatory(=)		
User data	Optional	Optional(=)		
Result			Mandatory	Mandatory(=)

(=): The value of the parameter is identical to the value of the corresponding parameter in the preceding request or response primitive.

1.4.5.2.1 Recovery point.

This parameter is used to enable the Master to indicate the recovery point which corresponds to the last data transfer primitive for which a positive confirmation had been received. The Transfer will be resumed from the data transfer service immediately following the indicated recovery point. The value of the recovery point is in the range 0 to 65535.

This parameter cannot be used if the recovery function has not been accepted at the time the Access regime was established.

1.4.5.2.2 Designation.

This parameter enables the Master to designate the information it wants to save. It is a byte sequence of no more than 254 bytes.

1.4.5.2.3 User data

This parameter is used to convey string type information of no more than 254 bytes.

1.4.5.2.4 Result

This parameter enables the Slave to indicate acceptance or refusal of the save request.

This parameter may take one of the following values:

a) acceptance,

b) user refusal, reason:

reason not specified,
erroneous designation,
file already exist,
erroneous recovery point,
erroneous user data,
other reason.

1.4.6 Rename service.

1.4.6.1 Function.

The Rename service enables the Master to rename the designation of a file within the Slave. This service is not available in the Basic Kernel.

Only the Master may issue a T-RENAME Request, provided the Access regime is established and the use of T-RENAME has been accepted by the Slave at the time the Access regime was established.

1.4.6.2 Parameters.

The different primitives and parameters required by the Rename service are described in the following table.

Parameters	T-RENAME Request	T-RENAME Indication	T-RENAME Response	T-RENAME Confirm.
New name	Mandatory	Mandatory(=)		
Designation	Mandatory	Mandatory(=)		
User data	Optional	Optional(=)		
Result			Mandatory	Mandatory(=)

(=): The value of the parameter is identical to the value of the corresponding parameter in the preceding request or response primitive.

1.4.6.2.1 New name.

This parameter provides the new designation for the required information. It is a byte sequence of no more than 254 bytes.

1.4.6.2.2 Designation.

This parameter enables the Master to designate the information it requires to be renamed. It is a byte sequence of no more than 254 bytes.

1.4.6.2.3 User data

This parameter is used to convey string type information of no more than 254 bytes.

1.4.6.2.4 Result

This parameter enables the Slave to indicate acceptance or refusal of the rename request.

This parameter may take one of the following values:
a) acceptance,

b) user refusal, reason:

reason not specified,
erroneous designation,
erroneous new name,
unknown file,
new name already in use,
erroneous user data,
other reason.

1.4.7 Delete service.

1.4.7.1 Function.

The Delete service enables the Master to delete a file within the Slave. This service is not available in the Basic Kernel.

Only the Master may issue a T-DELETE Request, provided the Access regime is established and the use of T-DELETE has been accepted by the Slave at the time the Access regime was established.

1.4.7.2 Parameters.

The different primitives and parameters required by the Delete service are described in the following table.

Parameters	T-DELETE Request	T-DELETE Indication	T-DELETE Response	T-DELETE Confirm.
Designation	Mandatory	Mandatory(=)		
User data	Optional	Optional(=)		
Result			Mandatory	Mandatory(=)

(=): The value of the parameter is identical to the value of the corresponding parameter in the preceding request or response primitive.

1.4.7.2.1 Designation.

This parameter enables the Master to designate the information it requires to be deleted. It is a byte sequence of no more than 254 bytes.

1.4.7.2.2 User data

This parameter is used to convey string type information of no more than 254 bytes.

1.4.7.2.3 Result

This parameter enables the Slave to indicate acceptance or refusal of the delete request.

This parameter may take one of the following values:

a) acceptance,

b) user refusal, reason:

reason not specified,
erroneous designation,
unknown file,
erroneous user data,
other reason.

1.4.8 Typed data service.

1.4.8.1 Function.

The Typed data service enables the transfer of information either from the Slave or from the Master to the other entity, provided the Access regime is established and the use of T-TYPED DATA has been accepted at the time the Access regime was established. This service is not available in the Basic Kernel.

1.4.8.2 Parameters.

The different primitives and parameters required by the Typed data service are described in the following Table.

Parameter	T-TYPED-DATA Request	T-TYPED-DATA Indication
user data	Mandatory	Mandatory(=)

(=): The value of the parameter is identical to the value of the corresponding parameter in the preceding request or response primitive.

1.4.8.2.1 User data

This parameter is used to transmit information from the user of application transfer service and is a sequence of no more than 254 bytes.

1.5 TRANSFER REGIME CONTROL.

If the selected service class is the symmetrical class and if the Basic Transfer Mode has not been selected at the Access regime establishment and provided no other Transfer regime is established, the Transfer regime is established by the Master within the Access regime:

a) on the acceptor's side upon transmission of one of the following primitives T-DIRECTORY Response (positive), T-LOAD Response (positive), or T-SAVE Response (positive),

b) on the initiator's side upon reception of one of the following primitives T-DIRECTORY Confirmation (positive), T-LOAD Confirmation (positive), or T-SAVE Confirmation (positive).

If the selected service is the symmetrical class and if the Basic Transfer Mode has been selected at the Access regime establishment, and no other Transfer regime is established, the Transfer regime is established by the Master within the Access regime:

a) on the initiator's side upon transmission of a T-WRITE (T-WRITE END in the case of a "short" file) Request primitive,

b) on the acceptor's side upon reception of a T-WRITE (T-WRITE END in the case of a "short" file) Indication primitive.

If the selected service class is the basic kernel and no other Transfer regime is established, the Transfer regime is established by the Master within the Association regime:

a) on the initiator's side upon transmission of a T-WRITE (T-WRITE END in the case of a "short" file) Request primitive,

b) on the acceptor's side upon reception of a T-WRITE (T-WRITE END in the case of a "short" file) Indication primitive.

The Transfer regime is terminated upon transmission of a T-WRITE-END Response or a T-U-EXCEPTION Request (reject) primitive and upon reception of T-WRITE-END Confirmation, T-U-EXCEPTION Indication (reject) or T-P-EXCEPTION Indication.

1.5.1 Mass transfer

This service provides for confirmed and reliable transfer of data to virtual memory space in the Receiver.

The virtual memory is regarded as a sequence of bytes or as a sequence of records. The relation of this virtual memory to files, foreground memory, or other devices in the actual receiving equipment is defined in each case by the file attributes included in the file header.

1.5.1.1 Function

The T-WRITE service carries data from the required file. The data associated with each of these primitives is intended to immediately follow that of the previous T-WRITE in the virtual memory space.

The T-WRITE-END service conveys the last set of data of the required file. It is a confirmed service. A positive confirmation of this service terminates the Transfer regime.

Only the Sender may issue a T-WRITE Request or a T-WRITE-END Request primitive.

If the selected service class is the basic kernel only the Master may become the Sender.

T-WRITE is an optionally confirmed service. A recovery point is provided with each T-WRITE primitive if an explicit confirmation is requested.

1.5.1.2 Parameters for T-WRITE.

Parameter	T-WRITE Request	T-WRITE Indication	T-WRITE Response	T-WRITE Confirm.	S
Explicit conf.	Mandatory	Mandatory(=)			
First block	Mandatory	Mandatory(=)			
Block number	Optional	Optional(=)	Optional	Optional(=)	S
Data field	Mandatory	Mandatory(=)			
Result			Mandatory	Mandatory(=)	

(=) :The value of the parameter is identical to the value of the corresponding parameter in the preceding request or response primitive.
S: symmetrical service only.

1.5.1.2.1 Explicit confirmation.

This parameter is used to indicate whether or not explicit confirmation is requested.

This parameter must always be set to "explicit confirmation requested" as soon as any given entity suggest an anticipation window other than 1 or the recovery function.

1.5.1.2.2 First block.

This parameter is used to indicate whether or not the data contained in the data field parameter consist in the beginning of the file.

1.5.1.2.3 Block number.

In the basic kernel this parameter is absent.

In the symmetrical service, the block number is mandatory if the recovery mechanism is selected or the anticipation window is not 1. In this case every block is numbered and the block number is increased by 1 at each transmission of a new block. The block number is in the range 0 to 65535. The first block is 0.

If a block number is present in a request, it must be sent back in the response.

1.5.1.2.4 Data field.

It is used to convey data from the file, it is a variable length string of no more than 1024 bytes in the basic kernel, and the maximum length is negotiated in the symmetrical service during the establishment of the Access regime.

1.5.1.2.5 Result.

This parameter indicates, if the explicit confirmation has been requested, that the service has been accepted or rejected. The result parameter set to "reject" has the effect of restarting the transmission from the T-WRITE following the last confirmed T-WRITE (or from the first T-WRITE of the file if no T-WRITE was previously confirmed).

1.5.1.3 Parameters for T-WRITE-END.

The T-WRITE-END service is a confirmed service, so the corresponding response code should be sent by the receiver before any other protocol exchange.

Parameter	T-WRITE-END Request	T-WRITE-END Indication	T-WRITE-END Response	T-WRITE-END Confirm.	S
First block	Mandatory	Mandatory(=)			
Block number	Optional	Optional(=)	Optional	Optional(=)	S
Data field	Optional	Optional(=)			
Result			Mandatory	Mandatory(=)	

(=): The value of the parameter is identical to the value of the corresponding parameter in the preceding request or response primitive.

1.5.1.3.1 First block.

This parameter is used to indicate whether or not the data contained in the data field parameter consist in the beginning of the file.

1.5.1.3.2 Block number.

In the basic kernel this parameter is absent.

In the symmetrical service, the block number is mandatory if the recovery mechanism is selected or the anticipation window is not 1. The block number is in the range 0 to 65535. The first block is 0.

If a block number is present in a request, it must be sent back in the response.

1.5.1.3.3 Data field.

It is used to convey data from the file, it is a variable length string of no more than 1024 bytes in the basic kernel, and the maximum length is negotiated in the symmetrical service during the establishment of the Access regime.

1.5.1.3.4 Result.

This parameter indicates:

- in the basic kernel, file acceptance or refusal,

- in the symmetrical service, this parameter indicates file acceptance or refusal, or rejection of the service(s) since the last confirmed recovery point.

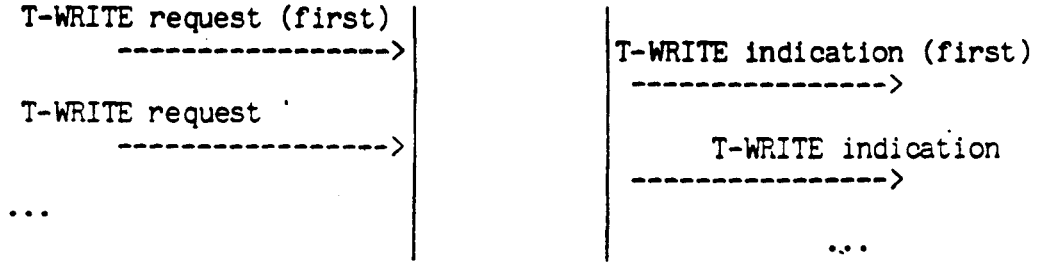
1.5.1.4 Operation of the mass transfer procedure

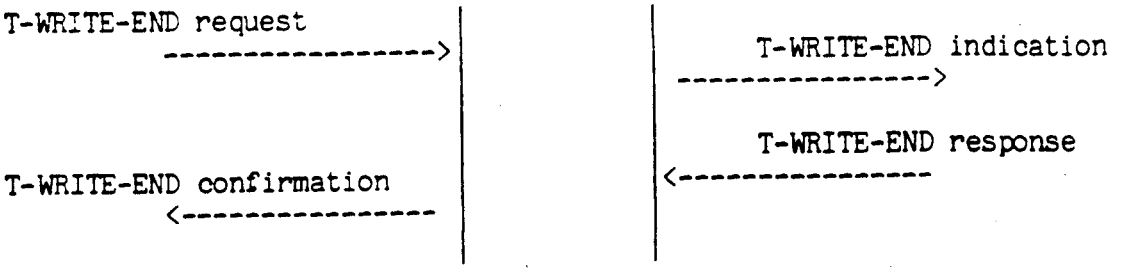
It is the purpose and the responsibility of the mass transfer phase to ensure the correct transfer of the identified mass data.

A mass transfer phase is initiated by T-WRITE request and indication primitives with the first block of data containing the file header or a part of it, and during a mass transfer phase, the reception of a valid T-WRITE indication should cause the data to be added or overwritten at the appropriate place in the defined memory space. The data fields accompanying T-WRITE or T-WRITE-END are assembled into a file according to the parameters specified in the file header, and a T-WRITE-END response and confirmation should only be transmitted when the receiver has successfully stored (as required) the transferred file. Positive confirmation of T-WRITE-END indicates that the Receiver accepts the responsibility of the file; negative confirmation indicates that the Receiver does not accept the responsibility of the file.

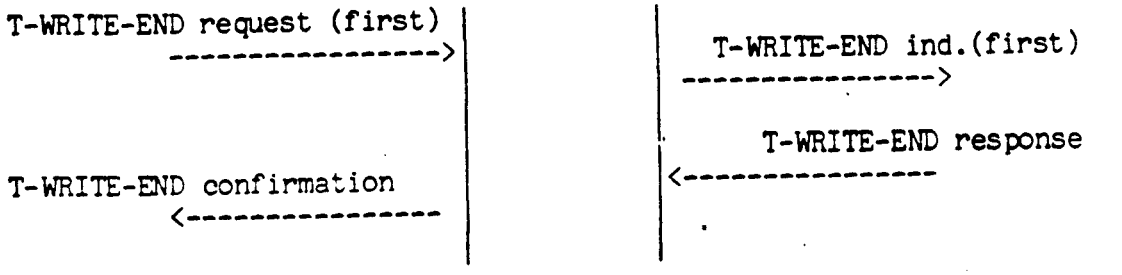
In the printer device application, the data fields are stored or passed to the printer device. The data must be transferred sequentially and contiguously. The T-WRITE-END response and confirmation should be issued only when the receiver has successfully received and forwarded the transferred data or has reliably stored it for later transfer.

In case the transfer is correctly completed, the sequence of events may be represented as follows:

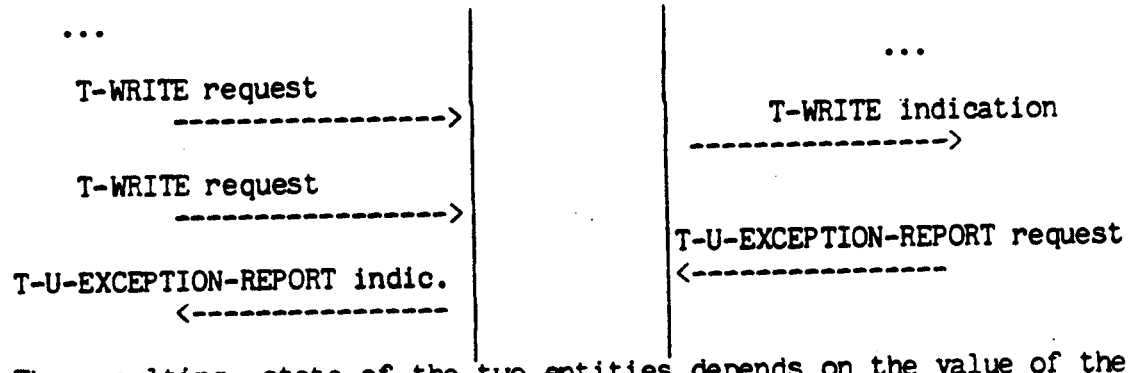




or



In case of an user error, the sequence of events is as follows:



The resulting state of the two entities depends on the value of the parameter "Reason" in the T-U-EXEPTION REPORT as defined in section 1.5.2.2.1.

1.5.2 Exception report service.

1.5.2.1 Function

This service is used in the Transfer regime:

by the Receiver to force the transfer abort or to restart the mass transfer at the beginning of the file.
by the Sender or the Receiver in the symmetrical service to force the transfer abort.

This service is performed by the T-U-EXCEPTION REPORT Request primitive.

1.5.2.2 Parameters

The different primitives and parameters required by the Exception report service are described in the following Table.

Parameter	T-U-EXC-REP. Request	T-U-EXC-REP. Indication
Reason	Mandatory	Mandatory (=)

(=): The value of the parameter is identical to the value of the corresponding parameter in the preceding request primitive.

1.5.2.2.1 Reason.

This parameter may take the following values:

a) "read restart":if used during the mass transfer phase it will cause the mass transfer to be reset and restarted from the beginning.

b) "transfer reject":used if the Receiver cannot support elements required by the file attributes transmitted in the first T-WRITE primitive, or in the case of any user error, when receiving a T-WRITE, to discontinue the transfer.

If this parameter is equal to "Transfer reject", the Transfer regime is terminated.

If this parameter is equal to "Read Restart" the Transfer regime is not interrupted and the transfer phase is resumed from its beginning.

1.5.2.3 Error recovery operation

1.6 EXCEPTION.

1.6.1 **Exception reporting.**

1.6.1.1 Function.

The T-P-EXCEPTION-REPORT Indication primitive is used by the service provider to signal exceptions during the service to the user on the other end. This primitive may only be used within the Access regime.

The exception is an error the service provider considers as recoverable.

After reception of T-P-EXCEPTION-REPORT the two entities are back in the idle state in the Access regime.

1.6.1.2 Parameters.

The different primitives and parameters required by the exception reporting service are described in the following table.

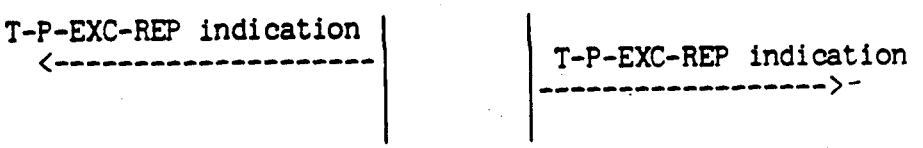
Parameter	T-P-EXC-REP. Indication
Reason	Mandatory

1.6.1.2.1 Reason

a) Reason:

reason not specified,
repeated negatives acknowledgments,
protocol conflict,
delay expired,
syntax error/missing parameter,
primitive not handled,
other reason.

1.6.1.3 Error recovery operation.



1.7 COLLISIONS.

In a given regime a user can normally issue a request only in the idle state, meaning no other request or indication is in the process.

In certain cases, both the entities are allowed to initiate services. Collisions between two services requested can thus arise.

1.7.1 **Collision in Association phase.**

In case of collision in the Association phase, the initiator of the physical connection ignores the T-Associate indication, and the acceptor of the physical connection must answer to the T-Associate indication so that only the association requested by the initiator is considered.

1.7.2 **Collision in the Association regime.**

The only possible collisions in the Association regime are the ones between a release request and an access or abort request.

The abort service has the highest priority, in this case the collision results always in the service abort. If an abort request collides with a service indication, the service indication is not taken into account and the abort request is carried out. If an abort indication collides with a service request, the service request is ignored and the abort indication is carried out.

The release service has priority on an access request; in case a collision occurs it will result in this case in an association termination.

If a release request collides with an access service indication, the service indication is not taken into account and the release request is carried out. If a release indication collides with an access service request, the access service request is ignored and the release indication is carried out.

If a release request collides with a release indication, the release indication of the Master is carried out and the other one is ignored.

1.7.3 Collision in Access regime.

In the Access regime a certain number of collisions between service requests can occur, these collisions are solved in the following way:

The following figure specifies the result of collisions in the Access regime.

Master / Slave	ABORT	END-ACCESS	DLSRD	TYPED-DATA	P-EXC-REP
ABORT	Abort	Abort	Abort	Abort	Abort
END-ACCESS	Abort	End-Access(1)	End-Access	End-Access	End-Access
TYPED-DATA	Abort	End-Access	2	3	3
P-EXC-REP	Abort	End-Access	2	3	4

ABORT = T-U-ABORT, T-P-ABORT,
END-ACCESS = T-END-ACCESS,
DLSRD = T-DIRECTORY, T-LOAD, T-SAVE, T-RENAME, T-DELETE,
T-TYPED-DATA = T-TYPED-DATA,
P-EXC-REP = T-P-EXC-REP.

Collision result:

Abort : the abort request is fulfilled. The association is terminated, data may be lost.

End-access : the end-access request is fulfilled. the Access regime is terminated. The other service is ignored or given up.

(1) : the Master's end-access request is fulfilled. The Slave's request is ignored or given up.

(2) : the Slave's request is fulfilled. The Master's request is confirmed normally by the Slave. The T-TYPED-DATA primitive has no consequence on the service.

(3) : the two requests are fulfilled.

(4) : no collision, back to the idle state in the Access regime.

1.7.4 Collision in the Transfer regime.

During the transfer phase, the priority order is the following: first the abort service then, the user's exception report service and the provider's exeption report service.

Whenever the service provider issues a T-P-ABORT Indication or a T-P-EXCEPTION REPORT Indication he gives up the current service and goes back to the idle state, as before the Association regime establishment or in the idle state in the Access regime.

Whenever the service provider receives a T-U-ABORT Request or a T-U-ABORT Indication, he gives up the current service and fulfills the user's abort service, and goes back to the idle state, as before the association regime establishment.

Whenever a transfer request and a abort transfer request collide, the abort transfer request is fulfilled, and all the transfer primitives in the process are given up.

2. TELESOFTWARE AND AUXILIARY DEVICE APPLICATIONS

2.1. PRELIMINARIES.

Two applications are hereby defined: telesoftware and auxiliary device.

Through a network, a telesoftware application allows interconnection of two machines and thus allows file exchange.

The telesoftware and auxiliary device application are defined by rules specifying the use of the service described in section 1 and by a virtual file structure which contains the characteristics of each virtual file and the relationship of these characteristics to the actions performed by the applications which manipulate these files.

A virtual file is a structured set of informations containing:

- a unique file name, that allows it to be referenced without ambiguity,
- descriptive file attributes which express characteristics of the file (date...),
- attributes describing the logical structure and the data stored in the file,
- data, forming the content of the file.

The three first kinds of information constitute the header of the file, and the data constitute the content of the file.

To be able to use the file transfer service, a practical implementation must state the relationship between the elements in the virtual file and the real storage system available.

In this application the files have been divided up in three different groups.

In the symmetrical service, the application does not make any assumption on the two machines and their functioning. This application may be used between two Servers, or two Executors, or between a Server and an Executor. Either entity may take the part of the Master depending on its requirements by redefining an Access regime within a given Association regime.

2.2. THE TELESOFTWARE APPLICATION ORGANIZATION.

The Telesoftware application uses files described in 2.5 and these files are transfered according to the rules described in 2.6.

The Telesoftware organization allows file exchange from the three different groups:

- group of transferable files (group A)
- group of application presentation files (group B)
- group of service support files (group C)

Note 2.1: When using the service class "basic kernel" only group A files are considered. In the symetrical service class, the default group is group A and other groups may not be implemented in some Data Bases.

2.2.1. Transferable files - group A -.

This group contains a data base of transferable files consisting of description, text, software, data and command files which together make up the application (see section 2.4).

These files can be obtained by their transfer name. The list of all transfer names make up the file's directory. The sublist of all transfer names of structure files make up the application's directory.

2.2.2. Application presentation file - group B -.

This group contains a data base of files describing the applications. If these files exist, they share the same transfer name as the corresponding description file (name of the application). The list of all the transfer names of application presentation files make up the application presentation file's directory.

These files are text files. They carry technical, user and managerial informations on the application sharing the same name. The aim of these files is to provide the maximum of information on the application in order to better guide the user.

2.2.3. Service support - group C -.

This group contains a data base for general information. These informations are organized by use of files and are accessible using key words. The list of all the transfer names of the service support files make up the service support file's directory.

These files are text files. They carry general information on the service.

2.2.4. Working area.

Several working areas are possible. It is up to the higher level application to define the different working areas. These applications should be able to offer access to a public and a private working area. The access rights to these working areas may depend on different criteria (example: password).

However access to one or the other working areas should be left transparent for the operator.

2.3 PRINTER APPLICATION ORGANIZATION

The printer device application is a specific case of the auxiliary device application. It uses files as described in 2.5. These files are transfered according to the rules described in 2.6. Not all the parameters of the file are used for the printer device application (see 2.5.1.17).

2.4. FILES.

2.4.1. File identification.

2.4.1.1 Preliminaries

In telesoftware, they are five types of files:

- description files (DeF),
- software files (SF),
- data files (DF),
- command files (CF),
- text files (TF).

The following sections indicate how these different files are regrouped in the three groups previously defined.

2.4.1.2. Description files.

These files belong to the group A. They carry data required to transfer the application, and also the names of all the files which are part of this application.

2.4.1.3 Software file.

These files belong to the group A. They carry software to be executed immediately or after processing.

2.4.1.4 Data files.

These files belong to the group A. They carry information required by software files at execution time.

2.4.1.5 Command files.

These files belong to the group A. They carry processing indications meaningfull for the entity who required transfer of these files. These indications are to be applied on files part of the application. The processing is done before or during execution of these files.

2.4.1.6 Text files.

These files may belong to the group A,B or C. These files may be created upon a directory request on group A,B or C.

2.4.1.6.1 Text files - group A -.

These files can be displayed and represent a user's manual or some data displayed during the execution of the application.

2.4.1.6.2 Text files - group B -.

These files carry displayable data. This data can be useful, before requiring transfer of the corresponding application.

2.4.1.6.3 Text files - group C -.

These files carry displayable data of the telesoftware support service.
This information is not necessarily related to the transferable applications.

2.4.1.6.4 Text files from a file directory request.

These files list all the file names belonging to the set of files where the request took place. The length of this list depends on the parameter designation in the file directory request.

2.4.2 Transferable applications.

A transferable application is a set of files. These files are part of the group A and their number is not limited.

2.4.2.1 Structure of a transferable application.

A transferable application is made up of a description file and at least one software, command, data or text file.

2.4.2.1.1. Purpose of the description file.

The purpose of the description file is to provide coded information regrouping the characteristics of a transferable application and the list of all files which are part of this application.

2.4.2.1.2. Organization of the description file.

Like all other files, the description file is made up of a header and of a main body.

The header is described in section 2.5.1.

The main body regroups all the headers of all the files which are part of this application.

The data in these headers are regrouped in each file.

This data provide useful elements needed to perform correctly the transfer of an application, for instance: type, transfer name, size, ressources, coding. Other data fields can be found.

2.4.3 File classification.

The files are classified into three groups.

Inside each group, the files have a unique transfer name. This name is kept in a directory. It points to an application whenever it is the name of a description file, and to a file in all other cases.

2.4.3.1 Structure of a transfer name .

A transfer name is made up of one or more keywords. The maximum number of keywords in a name is eight (8).

These keywords are called subnames of the transfer name.

2.4.3.1.1 Keywords.

A keyword is a byte sequence of no more than 12 bytes. Each byte can take different values.

2.4.3.1.2 Transfer name.

If a transfer name is made up of several keywords, then they are separated by a byte "/". The maximum length of a transfer name, counting all separators, is 70.

2.5 DESCRIPTION OF A VIRTUAL FILE.

A virtual file is a set of structured data having:

- the file's attributes: the header.
- the data in the file : the file's content.

This organization is valid for an application :

a) telesoftware,
b) printing,
c) telesoftware and printing.

2.5.1 Header

The header provides information on the characteristics of the file. It is always transmitted in the first T-WRITE (or T-WRITE-END for short files) and possibly in the following ones if necessary, in order to link the header information to the data. The file content may be transmitted in the first T-WRITE (if its data field is not filled up with the file header), and subsequently in other T-WRITE or T-WRITE-END.

The following attributes may be used:

2.5.1.1 File type

There are five different types of file:

- the description file which identifies the characteristics of the files which compound the application.
- the software files to be executed,
- the data files,
- the command files,
- the text files to be displayed or printed.

The default value is text file.

2.5.1.2 Execution order

This attribute indicates whether the application should be executed immediately or after the association to the telesoftware application has been released. It also indicates the starting file of the application. (default value: don't care).

2.5.1.3 Transfer name

This attribute represents the designation of the file on the sender's side (section 2.4.3.1.).

2.5.1.4 Filename

This attribute provides a "name" which may be associated with the transferred file in the filestore, (the filename is not necessarily related to any transfer name).

Note 2.2: Some systems may require that the first 6 characters of the filename should be alphabetic or numeric. Drive, device or directory designations should not be included in this attribute, but file-type suffices may be included.

2.5.1.5 Date/time of last modification

This attribute indicates the date and time of the last modification of the file.

2.5.1.6 File length

This attribute indicates the size in bytes of the file content after the downloading procedure.

2.5.1.7 Destination code

This attribute may take the values "don't care" (default), "foreground memory", "background memory random access required" (e.g. disk) or "cassette tape required".

Note 2.3: The latter option may be required for some software to overcome operating system or security constraints.

2.5.1.8 File coding

This attribute indicates the data syntax, the language, the machine type or the operating system.

According to the file type, this attribute can take the following values:

- Software file:
  - . Source code in a text form
  - . Source code in a tokenised form
  - . Intermediate code
  - . Object code
  - . Executable code (default value)
- Data file :
  - . Binary code (default value)
  - . Character code
- Command file:
  - . Dependant machine code (default value)
  - . Standardized code
- Text file:
  - . Character code (default value)
  - . Videotex code (+ profile)
  - . Other code

Additional information, if present, identify by means of a text string a language (such as "BASIC", "P-CODE") or a target processor (such as "6502"). This may optionally be followed by an identification of the language dialect (such as "MSDOS").

2.5.1.9 Destination name

This attribute indicates a drive, device and/or directory name for the storage of the file. It may not be used in combination with a destination code. The length of the attribute is limited to 255 bytes.

2.5.1.10 Cost

This attribute represents the price of the software, it is a string of no more than 31 bytes.The content of this field is only for information.

2.5.1.11 User field

This field is used to carry comments related to the file and consisting of no more than 254 displayable characters.

2.5.1.12 Load address

This attribute may be used to indicate an address in foreground memory at which the data should be loaded. In the case of a transfer to foreground memory, this attribute provides a base address for the transferred data, in the case of transfer to background memory, this attribute may be recorded as a file attribute according to the terminal filing's system. The relationship of this address to any executor operating system virtual memory addressing scheme is not specified by this protocol.

2.5.1.13 Execute address (absolute)

This attribute may be used to indicate an absolute address at which programme execution should start when the file is loaded into foreground memory. The relationship of this address to any executor operating system virtual memory addressing scheme is not specified by this protocol.

2.5.1.14 Execute address (relative)

This attribute may be used to indicate an address relative to the beginning of the file after loading into foreground memory, at which programme execution should start.

2.5.1.15 Compression mode

This attribute is used to indicate which compression algorithm is used for the file's content.

2.5.1.16 Device

This attribute is used to specify the characteristics of the device to which a file content is to be sent.

Note that in the basic kernel only the printer is taken into account.

2.5.1.17 Status of file attributes

FILE ATTRIBUTES	APPLICATION		DEFAULT VALUE
	Telesoftware	Printer	
File type	O	O	Text file
Execution order	O	-	Don't care
Transfer name	O	O	No
Filename	O	O	No
Date of last modif.	O	O	No
File length	O	O	No
Destination code	O	-	Don't care
File coding	O	O	Depends on file type
Destination name	O	-	No
Cost	O	O	No
User field	O	O	No
Load address (abs.)	O	-	No
Execute address (abs)	O	-	No
Execute address (rel)	O	-	No
Compression mode	O	O	No compression
Device	O	O	Don't care

O : Optional
- : Irrelevant

Note 2.4: Other attributes might be added to take into account the auxiliary device application requirements.

2.5.2 **File content**

The file's content carries executable data or data needed for presentation, downloading or execution of other files.

2.6 USE OF THE T-SERVICE FOR TELESOFTWARE AND PRINTER DEVICE APPLICATION.

2.6.1 Preliminaries.

The telesoftware and auxiliary device applications use all of the symmetrical class service primitives in the service.

Only one telesoftware application may be associated at a time, and only one file may be transmitted at a time.

All the attributes listed in section 2.5.1.17 may be used to characterize a file involved in a telesoftware application.

Only one printer device application may be associated at a time, and only one mass transfer may be processed at a time.

The attributes listed in section 2.5.1.17 may be used to characterize a file involved in a printer device application.

For telesoftware applications consisting in several files, a description file may be transmitted first. This file consists of its own header and contains all the headers of the files to be downloaded. Each header is self-delimited by TLV encoding. When transmitting the subsequent files, the transmitted headers may not include informations which were included in the description file.

This organization allows for using the TDU service for other file transfer applications than telesoftware which would require a different file organization (e.g. a specific application descriptor which would be different from the description file).

2.6.2 Association.

The purpose of the association is to establish a link between two applications.

The association uses the T-ASSOCIATE Request and the T-ASSOCIATE Response services primitives.

The application name will carry :

!T telesoftware,
!P printer's application,
!A telesoftware and printer's application.

If !T is used as an application name, only telesoftware files may be transfered during the association.

If !P is used as an application name, only printer device files may be transfered during the association.

If !A is used, printer device and telesoftware files may be exchanged during the association.

If the selected service class is "basic kernel", the restriction on the T-service apply and the access, end access, file directory, load, save help, rename, delete and typed data services are not available.

2.6.3 Release.

The release of the application is performed by the T-RELEASE Request and T-RELEASE Response service primitives.

2.6.4 Abort.

The abort service is used to leave an application in an abnormal way.

The T-U-ABORT Request primitive may be used by the application.

The parameter Reason in the Request/Indication indicates the reason for the abort.

2.6.5 Access.

The transfer conditions are established by use of the T-ACCESS Request and T-ACCESS Response service primitives.

If the role is Slave, the parameter User data indicates the scope of services like T-DIRECTORY, T-LOAD, T-SAVE, T-RENAME, T-DELETE on the groups A, B and C.

If the role is Master, the parameter User data is not used.

2.6.6 End of access.

Terminating an Access regime is performed by the T-END-ACCESS Request and T-END-ACCESS Response service primitives.

2.6.7 File directory.

The T-DIRECTORY Request primitive is used by the Master to request transfer of files, collection of file names and application name, from the Slave to the Master. In case the request is accepted, the file content will depend on the designation parameter in the request.

A directory request is performed by the T-DIRECTORY Request primitive.

The parameter Designation is a byte sequence indicating the criterias for selecting the file names.

The parameter User data indicates the group or subgroup on which applies the directory request.

The response/confirmation is provided by the parameter Result in the T-DIRECTORY Response primitive, this parameter indicates acceptance or the reason for the refusal.

Acceptance means that file or application names meet with the criteria in the request.

The Slave will then carry on with the transfer of the file listing all the file or application names fulfilling the request.

2.6.7.1 Byte sequence in a directory request.

The designation parameter in a directory request is made of one or more elementary words separated by a specific operator. Other operators as OR, AND, or specific codes as parenthesis or star can be used. A detailed description of the syntax is given in section 2.7.

The maximum number of elementary words is 8.

2.6.8 Load.

The load request is performed by the T-LOAD Request service primitive. This primitive is used by the Master to request file transfer from the Slave to the Master.

The designation parameter contains the file transfer name requested (see section 2.4.3.1).

The recovery parameter if it is present (provided it was previously allowed) indicates the number of the last T-WRITE correctly received. Recovery will start with the following number.

The user data parameter indicates the group required in the LOAD service. If this parameter is absent the group is meant to be the A group.

The response/confirmation is provided by the parameter Result in the T-LOAD Response primitive, this parameter indicates acceptance or the reason for the refusal.

Acceptance means that a file meets the request.

The Slave will then carry on with the transfer of the file.

2.6.9 Help.

Request of a help file can be performed by the T-LOAD Request service primitive on the group A, the value of the designation parameter is then 2/0.

The recovery parameter if it is present (provided it was previously allowed) indicates the number of the last T-WRITE correctly received. Recovery will start with the following number.

The response/confirmation is provided by the parameter Result in the T-LOAD Response primitive, this parameter indicates acceptance or the reason for the refusal.

Acceptance means that a file meets the request.

The Slave will then carry on with the transfer of the file.

The purpose of this file is to provide informations on the organization of the Slave's data base: structure, key words, file name...

This file is a text file coded characters.

2.6.10. Save.

In the basic kernel, the Master may request transfer from Master to Slave by initiating a mass transfer using T-WRITE.

In the symmetrical service, the T-SAVE Request primitive is used by the Master to request file transfer from the Master to the Slave.

In the telesoftware service, one can transfer files from any of the A, B or C groups (see 2.2).

The designation parameter contains the file transfer name requested (see section 2.4.3.1).

The recovery parameter if it is present (provided it was previously allowed) indicates the number of the last T-WRITE correctly received. Recovery will start with the following number.

The user data parameter indicates the group required in the SAVE service. If this parameter is absent the group is meant to be the A group.

The response/confirmation is provided by the parameter Result in the T-SAVE Response primitive, this parameter indicates acceptance or the reason for the refusal.

Acceptance means that a file meets the request.

The Master will then carry on with the transfer of the file.

2.6.11. Rename.

The T-RENAME Request primitive is used by the Master to request renaming of a file or application transfer name in the Slave's data base.

The designation and new name parameters contain the old and new file's (or application) transfer name.

The user data parameter indicates the group required in the RENAME service. If this parameter is absent the group is meant to be the A group.

The response/confirmation is provided by the parameter Result in the T-RENAME Response primitive, this parameter indicates acceptance or the reason for the refusal.

Acceptance means that a file was renamed.

Refusal indicates that the renaming was not performed.

2.6.12. Suppression.

The T-DELETE Request primitive is used by the Master to request the suppression of a file or application transfer name in the Slave's data base.

The designation parameter indicates the file to be deleted.

The user data parameter indicates the group required in the DELETE service. If this parameter is absent the group is meant to be the A group.

The response/confirmation is provided by the parameter Result in the T-DELETE Response primitive, this parameter indicates acceptance or the reason for the refusal.

Acceptance means that a file was deleted.

Refusal indicates that the suppression was not performed.

2.6.13. Transfer abort.

The T-U-EXCEPTION REPORT Request primitive is used by either entity to suspend or restart a file transfer.

The parameter indicates the reason for this.

2.7. THE DESIGNATION FIELD IN A DIRECTORY REQUEST.

The designation field in a directory request enables one to define a research criteria in the current data base.

A joker "*" can be used in this field to designate a character string. Also logical operators AND, displayed /, and OR, displayed +, can be used to combine various character strings. Parenthesis (and) indicate how these operators act.

Examples:

1) Any file in the data base meets criteria * or criterias * + *, */*, (*+*).

2) Files SOFT and SMALL.BAT meet criteria S*T.

3) Files SOFT, SMALL.BAT, SUN meet criteria S*/(*N + *T).

The aim of this annex is to define the syntax of the designation field in a directory request and the associated criteria research.

The sequences (AB), (A) + B*, (((*))), */* are all directory request syntactically correct, this is not the case of the following sequences: A++B, (((A*)B*)C*), A(/)B.

Definitions:

a) A **file name** is any sequence of no more than 8 key words separated by /, a **key word** is any sequence of no more than 12 bytes (any value between 2/1 and 7/14 is permitted with the exception of 2/8, 2/9, 2/10, 2/11, 2/15 displayed (,), *, + and /). Each one of these key words is called **subname** of the file name.

b) An **elementary word** is any sequence of no more than 12 bytes (any value between 2/1 and 7/14 is permitted with the exception of (,), + and /). Moreover no more than one byte is equal to *.

Using the above definitions, a correct syntax for a directory request can be specified by also using the following 4 grammatical rules ($ designates here an abstract symbol but not a character) :

```
(0)      $  --------> choose your elementary word
(1)      $  --------> ($)
(2)      $  --------> $ + $
(3)      $  --------> $/$

one reads   --------> changes into
```

Any sequence generated using those four rules is said to be a correct sequence for a directory request.

Example: S*/(*N + *T)

For this example the first rules (1), (2) and (3) are used:

```
$ ----------> $/$ ----------> $/($) ------------> $/($ + $)
      (3)              (1)                (2)
```

and then rule (0) is used to replace each symbol $ by an elementary word:

```
$/($ + $) -------> S*/($ + $) -------> S*/(*N + $)
```

and finally

```
-------> S*/(*N + *T)
```

To any byte sequence forming a correct directory request is associated a research criteria:

A file name is said to meet the criteria a if and only:

- if a is an elementary word and if substitution of the * (if any) by 0, 1 or more characters gives a subname of this file name.
  Examples: SOFT, RON/SOFT, SUN/RON/SOFT meet criteria SOFT.
  SOFT, SUN, SAND/SUN/ALL meet criteria S*.
  SUN, SAND/SUN/ALL meet criteria S*N.

- or if a can be written (b) and the file name meets criteria b.
  Examples: any file meets criteria (*).
  SOFT, SAND/SUN/ALL meet criteria (S*).

- or if a can be written b + c and the file name meets the criteria b OR criteria c.
  Examples: SOFT and SAND/SUN/ALL meet criteria *N + *T.
  SOFT and RON meet criteria (S*T) + (*N).

- or if a can be written b/c and the file name meets criteria b AND criteria c.
  Examples: SAND/SUN/ALL meets criteria S*D/A*.
  SOFT meets criteria S*/*T.
  RON/SOFT meets criteria *N/(SAND + R*).

Important: If a is not an elementary word, decomposing a in simpler criterias is made in this order. It is equivalent to say the operator / has higher priority than operator +.

Example: AB/T + CA is (AB/T) + CA.

3 T-PROTOCOL SPECIFICATION

3.1 DEFINITIONS

Block: A block of user information sent in a T-Write TDU.

Mass transfer phase: the mass transfer phase is started by sending (or receiving) the first T-Write TDU and terminated when receiving (or sending) a T-Abort , a T-Transfer-reject, a T-Read-restart or the confirmation to the last T-Write TDU .

TDU: Telesoftware Data Unit. These units are protocol elements which are used to handle the T-Protocol.

Idle state in the Association regime: The state which is reached when the association regime is established, when no other regime is established and when no other service is being initiated.

Idle state in the Access regime: The state which is reached when the access regime is established, when no Transfer regime is established and when no other service is being initiated.

Table 1 gives an overview of TDU's and their mapping on the service primitives. This table indicates which TDU must be sent for a given service primitive and which TDUs may be received in response to that TDU. However this table is not exhaustive and does not deal with collisions nor with the complete error handling mechanism.

TABLE 1: LIST OF THE TDU's USED

Issued service primitive	TDU	Possible response TDU	Received service primitive
T-ASSOCIATE request	T-Associate	T-Response-positive (S)	T-ASSOCIATE conf(+)
		T-Response-negative (S)	T-ASSOCIATE conf(-)
		T-Abort (A)	T-U/P-ABORT.ind
T-RELEASE request	T-Release	T-Response-positive (S)	T-RELEASE conf(+)
		T-Abort (A)	T-U/P-ABORT ind
T-U-ABORT request	T-Abort	No response is required	
T-ACCESS request	T-Access	T-Response-positive (S)	T-ACCESS conf(+)
		T-Response-negative (S)	T-ACCESS conf(-)
		T-Abort (A)	T-U/P-ABORT ind
T-END-ACCESS.req	T-End-Access	T-Response-positive (S)	T-END-ACCESS conf(+)
		T-Abort (A)	T-U/P-ABORT ind
T-DIRECTORY request	T-Directory	T-Response-positive (S)	T-DIRECTORY conf(+)
		T-Response-negative (S)	T-DIRECTORY conf(-)
		T-P-Exception (A)	T-P-EXCEPTION ind
		T-Abort (A)	T-U/P-ABORT ind
T-LOAD request	T-Load	T-Response-positive (S)	T-LOAD.conf(+)
		T-Response-negative (S)	T-LOAD.conf(-)
		T-P-Exception (A)	T-P-EXCEPTION ind
		T-Abort (A)	T-U/P-ABORT ind
T-SAVE request	T-Save	T-Response-positive (S)	T-SAVE conf(+)
		T-Response-negative (S)	T-SAVE conf(-)
		T-P-Exception (A)	T-P-EXCEPTION ind
		T-Abort (A)	T-U/P-ABORT ind
T-RENAME request	T-Rename	T-Response-positive (S)	T-RENAME conf(+)
		T-Response-negative (S)	T-RENAME conf(-)
		T-P-Exception (A)	T-P-EXCEPTION ind
		T-Abort (A)	T-U/P-ABORT ind

TABLE 1 (Continued): LIST OF THE TDU's USED

Issued service primitive	TDU	Possible response TDU	Received service primitive
T-DELETE request	T-Delete		
		T-Response-positive (S)	T-DELETE conf(+)
		T-Response-negative (S)	T-DELETE conf(-)
		T-P-Exception (A)	T-P-EXCEPTION ind
		T-Abort (A)	T-U/P-ABORT ind
T-WRITE request	T-Write		
		T-Response-positive (S)	T-WRITE conf(+)
		T-Response-negative (S)	T-WRITE conf(-)
		T-Transfer-reject (A1)	T-U-EXCEPTION ind
		T-Read-restart (A1)	T-U-EXCEPTION ind
		T-P-Exception (A)	T-P-EXCEPTION ind
		T-Abort (A)	T-U/P-ABORT ind
T-WRITE-END req.	T-Write(end)		
		T-Response-positive (S)	T-WRITE-END conf(+)
		T-Response-negative (S)	T-WRITE-END conf(-)
		T-Transfer-reject (A1)	T-U-EXCEPTION ind
		T-Read-restart (A1)	T-U-EXCEPTION ind
		T-P-Exception (A)	T-P-EXCEPTION ind
		T-Abort (A)	T-U/P-ABORT ind
T-TYPED-DATA req	T-Typed-data		
		No response is required	
		T-Abort (A)	T-U/P-ABORT ind
T-U-EXCEP- request (Read-rest)	T-Rd-restart		
		T-Write (S)	T-WRITE ind
		T-Abort (A)	T-U/P-ABORT ind
T-U-EXCEP- request (Any other reason)	T-Tr-reject		
		No response is required	
		T-P-Exception (A)	T-P-EXCEPTION ind
		T-Abort (A)	T-U/P-ABORT ind

(A) means that the TDU may be sent in an asynchronous manner (at any time after having received the TDU).
(A1) means that the TDU may be sent in an asynchronous manner (at any time after having received the TDU during the mass transfer phase).
(S) means that the TDU must be sent synchronously (in response to the other TDU.)
(+) the result parameter of the primitive is set to "accept"
(-) the result parameter of the primitive is set to "reject"

3.2 DESCRIPTION AND USE OF TDU

3.2.1 T-Associate

T-Associate is used to establish the association with a specified processable data application.

The response to a T-Associate is conveyed using the TDU T-Response-positive or T-Response negative.

3.2.1.1 Content of the T-Associate TDU and the associated responses

T-Associate	
Calling address	*
Called address	*
Application name	
Service class	
Explicit confirmation	
Timeouts	
Request identification	*
Identification	*
User data	
T-Response-positive	
Called address	*
Timeouts	*
Identification	*
User data	*
T-Response-negative	
Result	*

*: This parameter may only be present if the symmetrical service class is proposed.

The use of the parameters is described in the service definition.

The format of the T-Response positive indicates which service class is selected.

The Result parameter in T-Response-negative may indicate the following reasons:

- Reason not specified (default value)

- Rejection by the service provider:
  . Called address incorrect
  . Calling address incorrect
  . Service class refused

- Rejection by the service user:
  . Called address incorrect
  . Calling address incorrect
  . Service class refused
  . Application name unknown
  . Wrong identification

. Erroneous user data
. Other reason (This last value may be followed by a string of no more than 62 displayable characters)

3.2.1.2 Sending T-Associate

A T-ASSOCIATE request primitive results in sending a T-Associate TDU.

If no explicit confirmation have been requested, the sender is in the state of association established.

If explicit confirmation had been requested, the sender waits for a T-Response positive or negative. In this case no other action is permitted before reception of this response except aborting the association. The reception of this response results in this case in a T-ASSOCIATE confirmation with the appropriate result code. The association will be established as soon as a T-response positive is received. If a T-response-negative or a T-Abort is received, the association is not established and a D-U-Abort must be sent by the initiator.

Note 3.1: It is recommended that T-Abort is not used to reject a T-Associate for which an explicit confirmation was requested.

Note 3.2: If the symmetrical service class is proposed in T-Associate an explicit confirmation must be requested.

3.2.1.3 Receiving T-Associate

A valid incoming T-Associate TDU results in a T-ASSOCIATE indication primitive.

If no explicit confirmation is requested, the association is established (it may be rejected using T-Abort).

If explicit confirmation is requested, a T-ASSOCIATE response primitive results in sending a T-Response-positive or negative depending on the result parameter of the primitive. If a T-Response-positive is returned, the association is established, otherwise the association is not established.

3.2.2 T-Release

This TDU is used to request the orderly termination of the processable data application.

The response to a T-Release is conveyed using the TDU T-Response-positive.

3.2.2.1 Content of the T-Release TDU and the associated response

T-Release
User data

T-Response-positive
User data *

*: This parameter may only be present if the symmetrical service class has been selected.

The use of the parameters is described in the service definition.

3.2.2.2 Sending T-Release

A T-RELEASE request primitive results in a T-Release TDU. If the selected service class is "basic kernel", only the initiator of the association is permitted to send a T-Release TDU. A T-Release TDU may be sent at any time after the association has been established when no other regime is established.

No other action is permitted after having sent a T-Release TDU before having received a T-Response-positive TDU except aborting the association.

On reception of the T-Response-positive TDU a D-U-Abort must be sent. The reception of a T-Response-positive results in a T-RELEASE confirmation.

3.2.2.3 Receiving T-Release

A valid incoming T-Release TDU results in a T-RELEASE indication. A T-RELEASE confirmation primitive results in sending a T-Response-positive to acknowledge the T-Release, a negative response is not permitted.

3.2.3 **T-Abort**

This TDU is used to abruptly terminate the association.

3.2.3.1 Content of the T-Abort TDU

T-Abort
Reason *

*: This parameter may only be present if the symmetrical service class has been selected.

The following values of the Reason parameter are related to the T-P-ABORT service:
- Repeated negative acknowledgements / repeated errors
- Delay expired
- Unknown message
- Protocol conflict
- Unrecoverable lower layer error
- Syntax error
- Other reason (This last value may be followed by a string of no more than 62 displayable characters)

The following values of the Reason parameter are related to the T-U-ABORT service:
- Reason not specified (default value)
- Wrong identification
- Erroneous role/function
- Other reason (This last value may be followed by a string of no more than 62 displayable characters)

3.2.3.2 Sending T-Abort

A T-U-ABORT request primitive or an exception condition (see 3.3.2, 3.3.3) results in sending a T-Abort TDU.

A T-Abort may be sent at any time after having received a T-Associate, to reject a non confirmed T-Associate or to abruptly terminate an association.

If the selected service class is basic kernel, only the acceptor of the association is permitted to send a T-Abort.

3.2.3.3 Receiving T-Abort

A valid incoming T-Abort TDU results in issuing a T-U-ABORT or T-P-ABORT indication primitive, depending on the reason parameter. The association is terminated.

A D-U-Abort DDU must be sent on reception of a T-Abort TDU.

3.2.4 T-Access

T-Access is used to establish the Access regime. This TDU may only be used if the symmetrical service class has been selected.

The response to a T-Access is conveyed using the TDU T-Response-positive or T-Response negative.

3.2.4.1 Content of the T-Access TDU and the associated responses

T-Access
Role
Functions
Transfer unit size

Anticipation window
Recovery
Transfer Mode
User data

T-Response-positive
Functions
Transfer unit size
Anticipation window
Recovery
Transfer Mode
User data

T-Response-negative
Result

The use of the parameters is described in the service definition.

The Result parameter in T-Response-negative may indicate the following reasons:

- Reason not specified (default value)

- Rejection by the service provider:
  . Role refused

- Rejection by the service user:
  . Role refused
  . Insufficient primitives handled
  . Erroneous user data
  . Other reason (This last value may be followed by a string of no more than 62 displayable characters)

3.2.4.2 Sending T-Access

A T-ACCESS request primitive results in sending a T-Access TDU. Only the Master is permitted to send a T-Access TDU when the Association regime is established and provided that no other regime is established. Then the sender waits for a T-Response positive or negative; no other action is permitted before reception of this response except aborting the association. The reception of this response results in a T-ACCESS confirmation with the appropriate result code. The Access regime and the assigned role will be established as soon as a T-Response positive is received. If a T-Response-negative or a T-Abort is received, the Access regime is not established.

3.2.4.3 Receiving T-Access

A valid incoming T-Access TDU results in a T-ACCESS indication primitive.

A T-ACCESS response primitive results in sending a T-Response-positive or negative depending on the result parameter of the primitive. If a T-Response-positive is returned, the Access regime is established, otherwise the Access regime is not established.

3.2.5 T-End-Access

T-End-Access is used to request the termination of the Access regime.

The response to a T-End-Access is conveyed using the TDU T-Response-positive.

3.2.5.1 Content of the T-End-Access TDU and the associated response

T-End-Access
Reason
User data

T-Response-positive
User data

The use of the parameters is described in the service definition.

The Reason parameter in T-End-Access may take the following values:

- Termination of the access regime requested by the service user:
  . Reason not specified (default value)
  . User abort of the access regime
  . Insufficient primitives handled
  . Other reason (This last value may be followed by a string of no more than 62 displayable characters)

3.2.5.2 Sending T-End-Access

A T-END-ACCESS request primitive results in sending a T-End-Access TDU. A T-End-Access TDU may be sent at any time after the Access regime has been established when no other embedded regime is established.

No other action is permitted after having sent a T-End-Access TDU before having received a T-Response-positive TDU except aborting the association. Upon reception of a T-Response-positive, both entities are in an idle state with the association regime established.

3.2.5.3 Receiving T-End-Access

A valid incoming T-End-Access TDU results in a T-END-ACCESS indication. A T-END-ACCESS confirmation primitive results in sending a T-Response-positive to acknowledge the T-End-Access, a negative response is not permitted.

3.2.6 T-Directory

T-Directory is used to request the transmission of a file directory from the Slave to the Master and it establishes a Transfer regime.

The response to a T-Directory is conveyed using the TDU T-Response-positive or T-Response negative.

3.2.6.1 Content of the T-Directory TDU and the associated responses

T-Directory
Designation
User data

T-Response-positive

T-Response-negative
Result

The use of the parameters is described in the service definition.

The Result parameter in T-Response-negative may indicate the following reasons:

- Rejection by the service user:
  . Reason not specified (default value)
  . Erroneous designation
  . No answer to the request
  . Erroneous user data
  . Other reason (This last value may be followed by a string of no more than 62 displayable characters)

3.2.6.2 Sending T-Directory

A T-DIRECTORY request primitive results in sending a T-Directory TDU. Only the Master is permitted to send a T-Directory TDU when the Access regime is established and provided that no Transfer regime is established. Then the sender waits for a T-Response positive or negative; no other action is permitted before reception of this response except aborting the association or reporting an exception. The reception of this response results in a T-DIRECTORY confirmation with the appropriate result code. The Transfer regime will be established as soon as a T-Response positive is received. If a T-Response-negative, a T-P-Exception or a T-Abort is received, the Transfer regime is not established.

3.2.6.3 Receiving T-Directory

A valid incoming T-Directory TDU results in a T-DIRECTORY indication primitive.

A T-DIRECTORY response primitive results in sending a T-Response-positive or negative depending on the result parameter of the primitive. If a T-Response-positive is returned, the Transfer regime is established, otherwise the Transfer regime is not established.

3.2.7 T-Load

T-Load is used to request the transmission of a file From the Slave to the Master and it establishes a Transfer regime.

The response to a T-Load is conveyed using the TDU T-Response-positive or T-Response negative.

3.2.7.1 Content of the T-Load TDU and the associated responses

T-Load
Recovery point
Designation
User data

T-Response-positive

T-Response-negative
Result

The use of the parameters is described in the service definition.

The Result parameter in T-Response-negative may indicate the following reasons:

- Rejection by the service user:
  . Reason not specified (default value)
  . Erroneous designation
  . Unknown file
  . Erroneous recovery point
  . Erroneous user data
  . Other reason (This last value may be followed by a string of no more than 62 displayable characters)

3.2.7.2 Sending T-Load

A T-LOAD request primitive results in sending a T-Load TDU. Only the Master is permitted to send a T-Load TDU when the Access regime is established and provided that no Transfer regime is established. Then the sender waits for a T-Response positive or negative; no other action is permitted before reception of this response except aborting the association or reporting an exception. The reception of this response results in a T-LOAD confirmation with the appropriate result

code. The Transfer regime will be established as soon as a T-Response positive is received. If a T-Response-negative, a T-P-Exception or a T-Abort is received, the Transfer regime is not established.

3.2.7.3 Receiving T-Load

A valid incoming T-Load TDU results in a T-LOAD indication primitive.

A T-LOAD response primitive results in sending a T-Response-positive or negative depending on the result parameter of the primitive. If a T-Response-positive is returned, the Transfer regime is established, otherwise the Transfer regime is not established.

3.2.8 T-Save

T-Save is used to announce the transmission of a file from the Master to the Slave and it establishes a Transfer regime.

The response to a T-Save is conveyed using the TDU T-Response-positive or T-Response negative.

3.2.8.1 Content of the T-Save TDU and the associated responses

T-Save
Recovery point
Designation
User data

T-Response-positive

T-Response-negative
Result

The use of the parameters is described in the service definition.

The Result parameter in T-Response-negative may indicate the following reasons:

- Rejection by the service user:
  . Reason not specified (default value)
  . Erroneous designation
  . Already existing file
  . Erroneous recovery point
  . Erroneous user data
  . Other reason (This last value may be followed by a string of no more than 62 displayable characters)

3.2.8.2 Sending T-Save

A T-SAVE request primitive results in sending a T-Save TDU. Only the Master is permitted to send a T-Save TDU when the Access regime is established and provided that no Transfer regime is established. Then

the sender waits for a T-Response positive or negative; no other action is permitted before reception of this response except aborting the association or reporting an exception. The reception of this response results in a T-SAVE confirmation with the appropriate result code. The Transfer regime will be established as soon as a T-Response positive is reoeived. If a T-Response-negative, a T-P-Exception or a T-Abort is received, the Transfer regime is not established.

3.2.8.3 Receiving T-Save

A valid incoming T-Save TDU results in a T-SAVE indication primitive.

A T-SAVE response primitive results in sending a T-Response-positive or negative depending on the result parameter of the primitive. If a T-Response-positive is returned, the Transfer regime is established, otherwise the Transfer regime is not established.

3.2.9 T-Rename

T-Rename is used to change the designation of a file in the Slave's file system.

The response to a T-Rename is conveyed using the TDU T-Response-positive or T-Response negative.

3.2.9.1 Content of the T-Rename TDU and the associated responses

T-Rename
New name
Designation
User data

T-Response-positive

T-Response-negative
Result

The use of the parameters is described in the service definition.

The Result parameter in T-Response-negative may indicate the following reasons:

- Rejection by the service user:
  . Reason not specified (Default value)
  . Erroneous designation
  . Erroneous new name
  . Unknown file
  . New name already in use
  . Erroneous user data
  . Other reason (This last value may be followed by a string of no more than 62 displayable characters)

3.2.9.2 Sending T-Rename

A T-RENAME request primitive results in sending a T-Rename TDU. Only the Master is permitted to send a T-Rename TDU when the Access regime is established and provided that no Transfer regime is established. Then the sender waits for a T-Response positive or negative; no other action is permitted before reception of this response except aborting the association or reporting an exception. The reception of this response results in a T-RENAME confirmation with the appropriate result code.

3.2.9.3 Receiving T-Rename

A valid incoming T-Rename TDU results in a T-RENAME indication primitive.

A T-RENAME response primitive results in sending a T-Response-positive or negative depending on the result parameter of the primitive.

3.2.10 T-Delete

T-Delete is used to request the deletion of a file in the Slave's file system.

The response to a T-Delete is conveyed using the TDU T-Response-positive or T-Response negative.

3.2.10.1 Content of the T-Delete TDU and the associated responses

T-Delete
Designation
User data

T-Response-positive

T-Response-negative
Result

The use of the parameters is described in the service definition.

The Result parameter in T-Response-negative may indicate the following reasons:

- Rejection by the service user:
  . Reason not specified (default value)
  . Erroneous designation
  . Unknown file
  . Erroneous user data
  . Other reason (This last value may be followed by a string of no more than 62 displayable characters)

3.2.10.2 Sending T-Delete

A T-DELETE request primitive results in sending a T-Delete TDU. Only the Master is permitted to send a T-Delete TDU when the Access regime is established and provided that no Transfer regime is established. Then the sender waits for a T-Response positive or negative; no other action is permitted before reception of this response except aborting the association or reporting an exception. The reception of this response results in a T-DELETE confirmation with the appropriate result code.

3.2.10.3 Receiving T-Delete

A valid incoming T-Delete TDU results in a T-DELETE indication primitive.

A T-DELETE response primitive results in sending a T-Response-positive or negative depending on the result parameter of the primitive.

3.2.11 T-Typed-data

T-Typed-data is used to transfer data in either direction, independently from the currently assigned roles.

There is no response required by the TDU layer entity.

3.2.11.1 Content of the T-Typed-data TDU

T-Typed-data
User data

The use of the parameter is described in the service definition.

3.2.11.2 Sending T-Typed-data

A T-TYPED-DATA request primitive results in sending a T-Typed-data TDU. Both entities are permitted to send a T-Typed-data TDU when the Access regime is established and provided that no Transfer regime is established.

3.2.11.3 Receiving T-Typed-data

A valid incoming T-Typed-data TDU results in a T-TYPED-DATA indication primitive.

3.2.12 T-Write

T-Write is used to carry data during a mass transfer phase.

The optional response to a T-Write is conveyed using the TDU T-Response-positive or T-Response negative.

3.2.12.1 Content of the T-Write TDU and the associated responses

T-Write
Explicit confirmation
First/last
Block number *
Data .

T-Response-positive
Block number *

T-Response-negative
Block number *

3.2.12.1.1 First/last

This parameter indicates if the data transmitted in the T-Write TDU consist in the first block of data or the last block or both or none.

This parameter is set to "first" when the parameter "first block" of the T-WRITE request or T-WRITE-END request primitive is set to "first".

This parameter is set to "last" when the TDU is issued subsequently to a T-WRITE-END request primitive.

3.2.12.1.2 Explicit confirmation

This parameter indicates whether or not a positive or negative confirmation is expected to the T-Write TDU.

In the case of the last block of data or when the proposed anticipation window size is greater than 1 or when the recovery function is in use, the value must be set to "explicit confirmation requested".

3.2.12.1.3 Block number

This parameter may only be present when the symmetrical service class has been selected. It must be present when the proposed anticipation window size is greater than 1 or when the recovery has been selected.

In T-Write this parameter indicates the number of the block, this number is increased by 1 at each transmission of a T-Write.

In T-Response-positive, this number indicates the block number of the acknowledged T-Write.

In T-Response-negative this number indicates the block number of the first T-Write to be retransmitted.

In T-Response positive or negative the block number parameter is present if and only if it was present in the corresponding T-Write.

3.2.12.1.4 Data

The T-Write TDU may contain a block of no more than 1024 octets of data if the Basic Kernel is used. In the symmetrical service the maximum size of a block is negotiated in the T-Access service.

3.2.12.2 Sending T-Write

A T-WRITE request or T-WRITE-END request primitive results in a T-Write TDU. This TDU is sent in a D-Data DDU. Only the Sender is permitted to send a T-Write TDU in the following cases:
- In the idle state of the Association regime if the basic kernel service class has been selected,
- In the idle state of the Access regime if the symmetrical service class has been selected and the Basic Transfer mode has been selected,
- After having sent a T-Response-positive to a T-Directory or T-Load TDU,
- After having received a T-Response-positive to a T-Save TDU,
- During a mass transfer phase.

If the Basic Kernel is beeing used or a window size of 1 is being used in the symmetrical service then, if explicit confirmation has been requested, the Sender waits for a T-Response-positive or negative, in this case no other action is permitted before reception of this response except aborting the association or reporting an exception.
If the window size is not equal to 1, then the sender may transmit up to the window size number of T-WRITE request or up to T-WRITE-END request before waiting for a T-Response except aborting the association or reporting an exception.
The reception of T-Response results in this case in a T-WRITE confirmation or T-WRITE-END confirmation primitive with the appropriate result code.

In the case of T-Response negative on a T-Write which was not indicating "last block", the transfer must be resumed from the first T-Write sent after the last T-Write for which a T-Response-positive had been received (or from the T-Write (first) if no T-Write was acknowledged since the beginning of the mass transfer phase).

3.2.12.3 Receiving T-Write

The mass transfer phase is initiated by the reception of a T-Write TDU with the parameter first/last indicating the beginning of the file.

A valid incoming T-Write TDU results in a T-WRITE indication or T-WRITE-END indication primitive. If explicit confirmation was requested, a T-response positive or negative must be sent in return.

Note 3.3: T-Write may also be rejected by T-Transfer-reject, T-Read-restart or T-P-Exception.

3.2.13 T-Transfer-reject

This TDU is used by the terminal to request the termination of a mass transfer.

3.2.13.1 Content

T-Transfer-reject
Reason

The Reason parameter is absent in case of user rejection, otherwise it contains a string of no more than 62 displayable characters (Other reason).

3.2.13.2 Sending T-Transfer-reject

A T-U-EXCEPTION-REPORT request with the "transfer reject" reason code results in a T-Transfer-reject TDU.
A T-Transfer-reject may be sent at any time during the mass transfer phase after having transmitted a T-Write. If the selected service class is the basic kernel, only the Receiver may send a T-Transfer-reject TDU.

3.2.13.3 Receiving T-Transfer-reject

A valid incoming T-Transfer-reject TDU results in a T-U-EXCEPTION-REPORT indication. The mass transferphase is terminated.

3.2.14 T-Read-restart

This TDU is used by the Receiver to request the termination of a mass transfer and causes the retransmision of data from the beginning of the file.

3.2.14.1 Content

This TDU contains no parameter

3.2.14.2 Sending T-Read-restart

A T-U-EXCEPTION-REPORT request with the "read restart" reason code results in a T-Read-restart TDU.
A T-Read-restart may only be sent by the Receiver, at any time during the mass transfer phase after having received a T-Write.

3.2.14.3 Receiving T-Read-restart

A valid incoming T-Read-restart results in a T-U-EXCEPTION-REPORT indication. The mass transfer is terminated. The Sender should restart the mass transfer from the beginning.

3.2.15 T-P-Exception

This TDU is used to abort any service in the Access regime.

3.2.15.1 Content

T-P-Exception
Reason

The Reason parameter may take the following values:
- Repeated negative acknowledgements
- Protocol conflict
- Syntax error/missing parameter
- primitive not handled
- Other reason (This last value may be followed by a string of no more than 62 displayable characters)

3.2.15.2 Sending T-P-exception

A T-P-Exception may only be sent when the Access regime is established after the detection of an error by the service provider (see 3.3). T-P-Exception may not be sent in the idle state of the access regime.

Sending a T-P-Exception results in a local T-P-EXCEPTION-REPORT indication. Then the sending entity enters the idle state of the Access regime.

3.2.15.3 Receiving T-P-exception

A valid incoming T-P-exception results in a T-P-EXCEPTION-REPORT indication. Then the receiving entity enters the idle state of the Access regime.

3.2.16 T-Response-positive

This TDU is used to acknowledge the TDUs for which a confirmation is required by this protocol specification (see 3.2.1 to 3.2.12)

3.2.16.1 Content

T-Response-positive
TDU code *
Other parameters *

*: This parameter may only be present if the symmetrical service class is selected.

The TDU code parameter indicates which TDU is acknowledged.

The other parameters depend on the TDU which is acknowledged (see 3.2.1.1 to 3.2.12.1)

3.2.16.2 Sending T-Response-positive

The conditions for sending a T-Response positive are described in sections 3.2.1.3 to 3.2.12.3.

3.2.16.3 Receiving T-Response-positive

The actions to be taken on the reception of T-Response-positive are described in sections 3.2.1.2 to 3.2.12.2.

3.2.17 T-Response-negative

This TDU is used to refuse the TDUs for which a confirmation is required by this protocol specification (see 3.2.1 to 3.2.12)

3.2.17.1 Content

T-Response-negative	
TDU code	*
Result	*

*: This parameter may only be present if the symmetrical service class is selected.

The TDU code indicates which TDU is refused.

The Result parameter contains a reason which depends on the TDU which is refused (see 3.2.1.1 to 3.2.12.1)

3.2.17.2 Sending T-Response-negative

The conditions for sending a T-Response negative are described in sections 3.2.1.3 to 3.2.12.3.

3.2.17.3 Receiving T-Response-negative

The actions to be taken on the reception of T-Response-negative are described in sections 3.2.1.2 to 3.2.12.2.

3.3 EXCEPTIONS AND TIMERS

3.3.1 Application response timer

This timer is used to control the maximum delay between sending a TDU corresponding to a service request and receving the TDU corresponding to the service confirmation (T-Response positive or negative) or any other event intended to terminate the service request (exception report, collision etc...).

The delay is fixed by the service user and controlled by the service provider. At the Association establishment each entity indicates the maximum delay it requires to answer to a request. This delay is controlled by the peer entity.

If the proposed service class is basic kernel, only the initiator of the association indicate the value of the application response timer.

If this timer expires recovery may be attempted using the procedures described in 3.3.2 or 3.3.3.

Note 3.4: It is the host's responsibility to ensure that this timer does not expire when started on a T-response-positive or negative. (In particular, user dialogue using non processable VPDE's should not take place within an association).

3.3.2 Abnormal termination of the mass transfer

The following error conditions detected by the Receiver during a mass transfer phase cause the transmission by the Receiver of a T-Read-restart or T-Transfer-reject.

- occurrence of a DDU exception condition,
- reception of any TDU other than T-Write,
- expiration of the application response timer.

If more than five successive occurrences of the same error are detected, the Receiver may either send a T-Abort or D-U-Abort.

A mass transfer phase may also be abnormally terminated by:

- transmission of a D-U-Abort DDU by either entity.
- transmission of a T-Abort, T-Transfer-reject, T-Read-restart T-P-Exception by either entity.

3.3.3 Errors outside a mass transfer phase

The following error conditions may occur outside a mass transfer phase:

- occurrence of a DDU exception condition,
- reception of a T-Write TDU,
- reception of an unknown or unexpected TDU,
- expiration of the application response timer.

In this case the entity may send a T-Abort or a T-P-Exception. If more than five occurrences of the same error are detected then the entity sends a D-U-Abort.

3.4 USE OF DDU LAYER

The T-Associate TDU must be conveyed by the D-set mode DDU.

Other TDU's are conveyed in D-Data DDUs. It is possible to concatenate several TDUs in the same D-DATA provided that only the last one is a TDU which requires confirmation at the TDU level and that the total length does not exceed the DDU maximum size.

The T-Responses to a TDU must never be concatenated with any other TDU.

When DDU modes A, B, D and E are used, T-Abort, T-Read-restart, T-Transfer-reject, T-Response-positive, T-Response-negative are sent without using any DDU.

When a symmetrical DDU mode is used, these TDUs are conveyed within D-Data DDUs and must never be concatenated with any other TDU.

If the symmetrical service class is proposed, a symmetrical DDU mode must be selected.

When a TDU is to be confirmed (T-Associate or T-Write with explicit confirmation requested, T-Write(last), T-Release, T-Directory, T-Load, T-Save, T-Rename or T-Delete), the confirmation flag must be set in the corresponding D-Data.

On indication of a DDU exception (unrecoverable error at the DDU level) the TDU layer may either try to recover (send a T-Read-restart, T-Transfer reject or T-P-Exception) or abort the TDU association. If more than five occurrences of the same error are detected then the entity sends a D-U-Abort.

4. CODING OF TDUs

4.1 CODING OF TDUs

The coding is a Type Length Value (TLV) encoding. Each TDU is identified by a Command Identifier and the total length of the TDU is indicated in a Length Indicator.

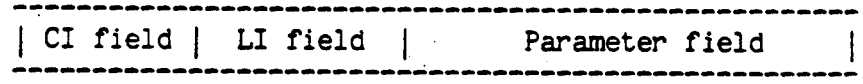
Note 4.1 : In the basic kernel the TDUs sent by the acceptor of the association only contain a Command Identifier without Length Indicator.

If the symetrical service class has been selected by the acceptor of the association all the TDU's contain at least a length indicator and may be followed by a parameter field (i.e. LI may be equal to 0).

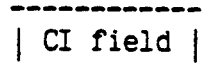
Note 4.2 : The whole TDU is subject to the 7-8 bit translation specified by the D-Set mode or by the D-Data conveying this TDU).

4.1.1 Structure of TDUs

The general structure of TDUs is the following:



or



4.1.1.1 Command Identifier field (CI)

The first byte of a TDU identifies the TDU according to the coding in Table 2.

4.1.1.2 Length Indicator field (LI)

The length indicator is used to indicate the length of the parameter field. The value of the LI field is expressed as a binary number representing the length in bytes of the associated parameter field (i.e. the value of the LI field does not include itself nor the CI field).

LI fields indicating lengths within the range 0-254 shall comprise one byte.
LI fields indicating lengths within the range 255-65534 shall comprise three bytes. The first byte shall be coded 15/15 and the second and third bytes shall contain the length of the associated parameter field with the high order bits in the first of these two bytes.

4.1.1.3 Parameter field

The parameter field comprises zero one or more groups of:

- Parameter Identifier field (PI): A single byte identifying the parameter. The coding of PIs is specified in Table 3.

- Length Indicator field (LI): This length indicator is used to indicate the length of the following parameter value. This LI is coded as specified in 4.1.1.2.

- Parameter Value field (PV): The coding of the parameter values is specified in section 4.1.2 below.

Note 4.3: The data field in T-Write only contains a PV field without any PI nor LI field.

The absence of a parameter in a parameter field means that either the default value applies if defined or that no information is associated with this parameter.

If a PV field is absent, the whole parameter group must be absent (i.e. LI may not be equal to 0).

In order to ensure easier migration to enhanced versions of the protocol the following rules should apply:

- parameter fields with undefined parameter identifiers should be ignored in reception.
- reserved bits in a parameter value are set to zero and ignored in reception.

Bits in a byte are numbered from b0 (least significant) to b7 (most significant).

4.1.2 Coding of TDUs

TABLE 2 - CODING OF TDU COMMAND IDENTIFIERS

TDU	CI	
T-Associate	2/0	
T-Release	2/1	
T-Abort	3/8	
T-Access	2/2	*
T-End-Access	2/3	*
T-Directory	2/4	*
T-Load	2/5	*
T-Save	2/6	*
T-Rename	2/7	*
T-Delete	2/8	*
T-Typed Data	2/9	*
T-Write	2/15	
T-Transfer-reject	3/6	
T-Read-restart	3/7	
T-P-Exception	2/14	*
T-Response positive	3/2	
T-Response negative	3/3	

*: symmetrical service only.

TABLE 3 - CODING OF TDU PARAMETER IDENTIFIERS

Parameter	PI	
User data	4/0	
Called address	4/1	*
Calling address	4/2	*
Result/Reason	4/3	*
Role/function	4/4	*
Application name	4/5	
Appl. resp. timeout	4/6	
Size/recovery/window	4/7	*
Designation	4/8	*
New name	4/9	*
Request identification	4/10	*
Identification	4/11	*
Explicit confirmation, first/last, block number	4/12	
Transfer mode	4/13	*
Recovery point	4/15	*
Service class	5/1	

*: symmetrical service only.

4.1.2.1 Associate

4.1.2.1.1 Request.

CI T-Associate = 2/0
LI = L

* PI Called address = 4/1
LI = n < 255
PV = n bytes (see 4.1.2.1.3)

* PI Calling address = 4/2
LI = n < 255
PV = n bytes (see 4.1.2.1.4)

PI Application name = 4/5
LI = n < 17
PV = n bytes (see 4.1.2.1.5)

PI App. resp timeout = 4/6
LI = 0/1
PV = 1 byte (see 4.1.2.1.6)

PI Service class = 5/1
LI = 0/1
PV = 1 byte (see 4.1.2.1.7)

PI Explicit Confirm. = 4/12
LI = 0/1
PV = 1 byte (see 4.1.2.1.8)

* PI Identification = 4/11
LI = n < 32
PV = n bytes (see 4.1.2.1.9)

```
*  PI Request identif.   = 4/10
   LI                    = 0/1
   PV                    = 1 byte (see 4.1.2.1.10)

   PI User data          = 4/0
   LI                    = n < 255
   PV                    = n bytes (see 4.1.2.1.11)
```

*: These parameters may only be present in the symetrical service class.

4.1.2.1.2 Responses

```
CI T-Response-pos.       = 3/2
LI                       = L

   PI Called address     = 4/1
   LI                    = n < 255
   PV                    = n bytes (see 4.1.2.1.3)

   PI Result             = 4/3
   LI                    = 0/1
   PV                    = 2/0

   PI App. resp timeout  = 4/6
   LI                    = 0/1
   PV                    = 1 bytes (see 4.1.2.1.6)

   PI Identification     = 4/11
   LI                    = n < 32
   PV                    = n bytes (see 4.1.2.1.9)

   PI User data          = 4/0
   LI                    = n < 255
   PV                    = n bytes (see 4.1.2.1.11)


CI T-Response-neg.       = 3/3
LI                       = L

   PI Result             = 4/3
   LI                    = n < 65
   PV                    = 2/0 + reason (see 4.1.2.1.12)
```

In the basic kernel, the responses are coded on 1 byte:

```
   T-Response positive   = 3/2
   T-Response negative   = 3/3
```

If the symetrical service class is proposed, the sender of T-Associate must be able to recognize the Response coding of both the basic kernel and the symmetrical service.

4.1.2.1.3 Called address.

This parameter is a variable length string of no more than 254 bytes. If used then it is the responsibility of the application to insert an appropriate identification in this field.

4.1.2.1.4 Calling address.

This parameter is a variable length string of no more than 254 bytes. If used then it is the responsibility of the application to insert an appropriate identification in this field.

4.1.2.1.5 Application name.

This parameter is a variable length string of no more than 16 bytes containing an application name. The use of initial character 2/1 ('!') is reserved for standardised applications.

4.1.2.1.6 Application response timeout.

This parameter is coded on 1 byte. The timeout interval is coded in binary form in increments of 1s.
The value 0/0 (default value) has the special meaning of 'timer disabled'

4.1.2.1.7 Service class.

This parameter is coded on 1 byte. The bit b0 is set to 1 to indicate that the basic kernel is used. The bit b1 is set to 1 to indicate that the symmetrical service is used. All other bits are reserved.

4.1.2.1.8 Explicit confirmation.

This parameter is coded on 1 byte.

Bits b0,b1,b2,b4,b5,b6,b7 are reserved.
b3 = 0 : explicit confirmation not requested.
b3 = 1 : explicit confirmation requested.
The default value is all bits set to 0 except b3 set to 1.
If the symmetrical service class is proposed, then bit b3 must be set to 1.

4.1.2.1.9 Identification.

This parameter is a variable length string of no more than 31 bytes. This string contains identification parameters separated by 2/15 (maximum length of each parameter is 12 bytes).

4.1.2.1.10 Request identification.

This parameter is coded on 1 byte.

b0 = 0 : no Identification requested in the response (default value),
b0 = 1 : an Identification is requested in the response.

All other bits are reserved.

4.1.2.1.11 User data.

This parameter is a variable length string of no more than 254 bytes.

4.1.2.1.12 Reason (in the Result parameter).

This value is a variable length string of no more than 63 bytes. The default value is: reason not specified. The coding of the reason values is specified in section 4.2.1.

4.1.2.2 T-Release.

4.1.2.2.1 Request

CI T-Release	= 2/1
LI	= L
PI User data	= 4/0
LI	= n < 255
PV	= n bytes (see 4.1.2.1.11)

4.1.2.2.2 Response.

CI T-Response-pos.	= 3/2
LI	= L
PI Result	= 4/3
LI	= 0/1
PV	= 2/1
PI User data	= 4/0
LI	= n < 255
PV	= n bytes (see 4.1.2.1.11)

In the the basic kernel, the response is coded on 1 byte: 3/2.

4.1.2.3 T-Abort.

4.1.2.3.1 Request .

CI T-Abort	= 3/8
LI	= L
PI Reason	= 4/3
LI	= n < 64
PV	= n bytes (see 4.2.2)

In the basic kernel, this TDU is coded on 1 byte : 3/8.

4.1.2.4 T-Access.

4.1.2.4.1 Request.

```
CI T-Access                = 2/2
LI                         = L

   PI Role/function        = 4/4
   LI                      = n < 3
   PV                      = 1 or 2 bytes (see 4.1.2.4.3)

   PI Size/recovery/win.   = 4/7
   LI                      = 0/1
   PV                      = 1 byte (see 4.1.2.4.4)

   PI Transfer mode        = 4/13
   LI                      = 0/1
   PV                      = 1 byte (see 4.1.2.4.5)

   PI User data            = 4/0
   LI                      = n < 255
   PV                      = n bytes (see 4.1.2.4.6)
```

4.1.2.4.2 Response.

```
CI T-Response-pos.         = 3/2
LI                         = L

   PI Result               = 4/3
   LI                      = 0/1
   PV                      = 2/2

   PI Role/function        = 4/4
   LI                      = n < 3
   PV                      = 1 or 2 bytes (see 4.1.2.4.3)

   PI Size/recovery/win.   = 4/7
   LI                      = 0/1
   PV                      = 1 byte (see 4.1.2.4.4)

   PI Transfer mode        = 4/13
   LI                      = 0/1
   PV                      = 1 byte (see 4.1.2.4.5)

   PI User data            = 4/0
   LI                      = n < 255
   PV                      = n bytes (see 4.1.2.4.6)


CI T-Response-neg.         = 3/3
LI                         = L

   PI Result               = 4/3
   LI                      = n < 65
   PV                      = 2/2 + reason (see 4.1.2.1.12)
```

4.1.2.4.3 Role/function.

This parameter is coded on 1 or 2 bytes.
The first byte is coded as follows:

b0 = 0 : Slave.
b0 = 1 : Master.

If the role is **Slave**, the other bits indicate the TDUs which may be received by this entity:

b1 = 1 : T-Read-restart.
b2 = 1 : T-Typed-data.
b3 = 1 : T-Directory.
b4 = 1 : T-Delete.
b5 = 1 : T-Rename.
b6 = 1 : T-Save.
b7 = 1 : T-Load.

If the role is **Master**, the other bits indicate the TDUs which may be received by this entity:

b1 = 1 : T-Read-restart.
b2 = 1 : T-Typed-data.

The other bits are reserved.

A second byte is reserved for future use and must not be sent in this version of the protocol.

No default value.

4.1.2.4.4 Size, recovery, window.

This parameter is coded on 1 byte. It indicates:

b0 = 0 : no recovery accepted by the Slave outside the mass transfer.
b0 = 1 : recovery accepted by the Slave outside the mass transfer.

This bit has no meaning if the role is Master.

b3, b2, b1 indicate the maximum size of the application data, contained in a T-Write or T-Write (last), accepted by the Receiver:

b3, b2, b1 =	000 : 512,	100 : 8192,
	001 : 1024,	101 : 16384,
	010 : 2048,	110 : 32768,
	011 : 4096,	111 : 65536.

b4 : reserved.

b7, b6, b5 indicate the maximum number of consecutive T-Write or T-Write (last) (Confirmation required) the Receiver can accept before issuing an answer:

b7, b6, b5 = 000 : 1, 100 : 5,
001 : 2, 101 : 6,
010 : 3, 110 : 7,
011 : 4, 111 : 8.

Default value: size: 1024 bytes, no recovery, window: 1.

4.1.2.4.5 Transfer mode

This parameter is coded on 1 byte as follows:

b0 = 0 : Basic Transfer mode not supported by Slave (if bit b0 of Role/function is set to 0)
Basic Transfer mode required by Master (if bit b0 of Role/function is set to 1)

b0 = 1 : Basic Transfer mode supported by Slave (if bit b0 of Role/function is set to 0)
Basic Transfer required by Master (if bit b0 of Role/function is set to 1).

b0 = 0 default value

All other bits are reserved.

4.1.2.4.6 User data.

For Telesoftware and/or Printer Device applications see section 4.4.1.

4.1.2.5 T-End-access.

4.1.2.5.1 Request.

CI T-End-access	= 2/3
LI	= L
PI Reason	= 4/3
LI	= n < 64
PV	= n bytes (see 4.2.2)
PI User data	= 4/0
LI	= n < 255
PV	= n bytes (see 4.1.2.1.11)

4.1.2.5.2 Response.

CI T-Response-pos.	= 3/2
LI	= L
PI Result	= 4/3
LI	= 0/1
PV	= 2/3

PI User data = 4/0
LI = n < 255
PV = n bytes (see 4.1.2.1.11)

4.1.2.6 T-Directory.

4.1.2.6.1 Request.

CI **T-Directory** = 2/4
LI = L

PI User data = 4/0
LI = n < 255
PV = n bytes (see 4.1.2.6.3)

PI Designation = 4/8
LI = n < 255
PV = n bytes (see 4.1.2.6.4)

4.1.2.6.2 Responses.

CI **T-Response-pos.** = 3/2
LI = L

PI Result = 4/3
LI = 0/1
PV = 2/4

CI **T-Response-neg.** = 3/3
LI = L

PI Result = 4/3
LI = n < 65
PV = 2/4 + reason (see 4.1.2.1.12)

4.1.2.6.3 User data.

For Telesoftware and/or Printer Device applications see section 4.4.2.

4.1.2.6.4 Designation

For Telesoftware and/or Printer Device applications see section 4.4.3.

4.1.2.7 T-Load.

4.1.2.7.1 Request.

CI T-Load	= 2/5
LI	= L
PI User data	= 4/0
LI	= n < 255
PV	= n bytes (see 4.1.2.7.5)
PI Designation	= 4/8
LI	= n < 255
PV	= n bytes (see 4.1.2.7.4)
PI Recovery point	= 4/15
LI	= n < 3
PV	= n bytes (see 4.1.2.7.3)

4.1.2.7.2 Responses.

CI T-Response-pos.	= 3/2
LI	= L
PI Result	= 4/3
LI	= 0/1
PV	= 2/5

CI T-Response-neg.	= 3/3
LI	= L
PI Result	= 4/3
LI	= n < 65
PV	= 2/5 + reason (see 4.1.2.1.12)

4.1.2.7.3 Recovery point.

This parameter is only present if this facility has been proposed and if the recovery is not performed from the beginning of the file.

The following byte(s) (maximum 2 bytes), if present, indicate(s) the block number.

The block number is coded in absolute binary form using all eight bits of each byte. It is coded on 1 byte between 0 and 255 and on 2 bytes between 256 and 65535. If 2 bytes are used then the first byte contains the most significant bits.
The recovery points are numbered consecutively and the first block is coded 0.

4.1.2.7.4 Designation.

For Telesoftware and/or Printer Device application see section 4.4.4.

4.1.2.7.5 User data.

For Telesoftware and/or Printer Device application see section 4.4.5.

4.1.2.8 T-Save.

4.1.2.8.1 Request.

CI T-Save	= 2/6
LI	= L
PI User data	= 4/0
LI	= n < 255
PV	= n bytes (see 4.1.2.7.5)
PI Designation	= 4/8
LI	= n < 255
PV	= n bytes (see 4.1.2.7.4)
PI Recovery point	= 4/15
LI	= n < 3
PV	= n bytes (see 4.1.2.7.3)

4.1.2.8.2 Responses.

CI T-Response-pos.	= 3/2
LI	= L
PI Result	= 4/3
LI	= 0/1
PV	= 2/6
CI T-Response-neg.	= 3/3
LI	= L
PI Result	= 4/3
LI	= n < 65
PV	= 2/6 + reason (see 4.1.2.1.12).

4.1.2.9 T-Rename.

4.1.2.9.1 Request.

CI T-Rename	= 2/7
LI	= L
PI User data	= 4/0
LI	= n < 255
PV	= n bytes (see 4.1.2.7.5)
PI Designation	= 4/8
LI	= n < 255
PV	= n bytes (see 4.1.2.9.3)

PI New name = 4/9
LI = n < 255
PV = n bytes (see 4.1.2.9.3)

4.1.2.9.2 Responses.

CI T-Response-pos. = 3/2
LI = L

PI Result = 4/3
LI = 0/1
PV = 2/7

CI T-Response-neg. = 3/3
LI = L

PI Result = 4/3
LI = n < 65
PV = 2/7 + reason (see 4.1.2.1.12)

4.1.2.9.3 Designation, New name.

The parameters Designation and New name contain the Transfer name and the New Transfer name of the file. (see 4.1.2.7.4 for coding).

4.1.2.10 T-Delete

4.1.2.10.1 Request.

CI T-Delete = 2/8
LI = L

PI User data = 4/0
LI = n < 255
PV = n bytes (see 4.1.2.7.5)

PI Designation = 4/8
LI = n < 255
PV = n bytes (see 4.1.2.7.4)

4.1.2.10.2 Responses.

CI T-Response-pos. = 3/2
LI = L

PI Result = 4/3
LI = 0/1
PV = 2/8

CI T-Response-neg. = 3/3
LI = L

PI Result = 4/3
LI = n < 65
PV = 2/8 + reason (see 4.1.2.1.12)

4.1.2.11 T-Typed data.

4.1.2.11.1 Request.

CI T-Typed data = 2/9
LI = L

PI User data = 4/0
LI = n < 255
PV = n bytes

4.1.2.12 T-Write.

4.1.2.12.1 Request

CI T-Write = 2/15
LI = L

PI Explicit confirm = 4/12
LI = n < 4
PV = n bytes (see 4.1.2.12.4)

PV Data field = N bytes

The Data field length is implicitely given as: N = L - (n + 2) bytes.

In the symmetrical service N is less or equal to the value negotiated in the T-Access service,
In the basic kernel N is less than 1025 bytes.

4.1.2.12.2 Responses to a T-Write (not last).

CI T-Response-pos. = 3/2
LI = L

PI Result = 4/3
LI = n < 4
PV = 2/15 + block number (see 4.1.2.12.5)

CI T-Response-neg. = 3/3
LI = L

PI Result = 4/3
LI = n < 4
PV = 2/15 + block number (see 4.1.2.12.5)

In the basic kernel, the response is coded on one byte 3/2 or 3/3.

4.1.2.12.3 Responses to a T-Write (last).

CI T-Response-pos. = 3/2
LI = L

PI Result = 4/3
LI = n < 4
PV = 2/15 + block number (see 4.1.2.12.5)

CI T-Response-neg. = 3/3
LI = L

PI Result = 4/3
LI = n < 67
PV = 2/15+ block number + reason (see4.1.2.12.5)

Symmetrical service:

- a T-Response positive means file acceptation.
- a T-Response negative means:
  if there is a user reason (erroneous file,other reason), file refusal.
  if there is no reason, last bloc(s) refusal.

Basic kernel:

The response is coded on 1 bytes 3/2 or 3/3.

- a T-Response positive means file acceptation.
- a T-Response negative means file refusal.

4.1.2.12.4 Explicit confirmation, first/last, block number.

This parameter is coded on 1 byte in the basic kernel, and 1 or more bytes in the symmetrical service.

The first byte contains the indication of explicit confirmation and the indication of first/last block.

Bits b7,b6,b5,b4,b2 are reserved

b3 = 0 : explicit confirmation not requested.
b3 = 1 : explicit confirmation requested.

b1,b0 = 0 0 : block
0 1 : first block
1 0 : last block
1 1 : first and last block

When b1 = 1 (last block) b3 must be set to 1.

The following byte(s) (maximum 2 bytes), if present, indicate(s) the block number.

The block number is coded in absolute binary form using all eight bits of each byte. It is coded on 1 byte between 0 and 255 and on 2 bytes between 256 and 65535. If 2 bytes are used then the first byte contains the most significant bits.
The recovery points are numbered consecutively and the first block is coded 0.

4.1.2.12.5 Result.

This parameter contains the TDU code and the block number if present in the T-Write with the same coding. It may also contain the reason of the negative response to the T-Write last.

4.1.2.13 T-Transfer-reject.

4.1.2.13.1 Request.

CI T-Transfer-reject	= 3/6
LI	= L
PI Reason	= 4/3
LI	= n < 64
PV	= n bytes (see 4.2.2)

In the basic kernel, this TDU is coded on 1 byte: 3/6.

4.1.2.14 T-Read-restart.

4.1.2.14.1 Request.

CI T-Read-restart	= 3/7

4.1.2.15 T-P-Exception.

4.1.2.15.1 Indication.

CI T-P-Exception	= 2/14
LI	= L
PI Reason	= 4/3
LI	= n < 64
PV	= n bytes (see 4.2.2)

4.2 CODING OF PROVIDER AND USER REFUSALS.

The reason codes are contained in the Result parameter of some TDUs to indicate either a user or a provider refusal. The following codes are allocated.. If the user reason is not specified then no code is transmitted.

4.2.1 Reason codes in a T-Response negative.

	provider	user	
Called address incorrect	3/0	4/0	
Calling address incorrect	3/1	4/1	
Role refused	3/2	4/2	
Insufficient primitives handled		4/3	
Application name unknown		4/4	
Service class refused	3/5	4/5	
Erroneous recovery point		4/6	
Erroneous designation		4/7	
No answer to the request		4/8	
Unknown file		4/9	
Already existing file		4/10	
Erroneous file (T-Write last)		4/11	
Erroneous new name		4/12	
New name already in use		4/13	
Wrong identity		5/0	
Erroneous user data		6/0	
Erroneous user data, service unknown		6/0, 6/1	
Erroneous user data, group forbidden		6/0, 6/2	
Other reason (+ optional string of char <63 bytes)		6/15	*

4.2.2 Reason in other TDUs.

4.2.2.1 Provider reason.

Repeated negative acknowledgements/ repeated errors	7/0	
Delay expired	7/1	
Unknown message	7/2	
Syntax error/ missing parameter	7/3	
Unrecoverable lower layer error	7/4	
Protocol conflict	7/5	
Primitive not handled	7/6	
Other reason (+ optional string of char <63 bytes)	6/15	*

4.2.2.2 User reason.

Wrong identification	5/0	
Role refused	4/2	
Insufficient primitives handled	4/3	
Other reason (+ optional string of char <63 bytes)	6/15	*

* : The string must contain displayable characters (codes 2/0 to 7/14 of the videotex primary graphic set)

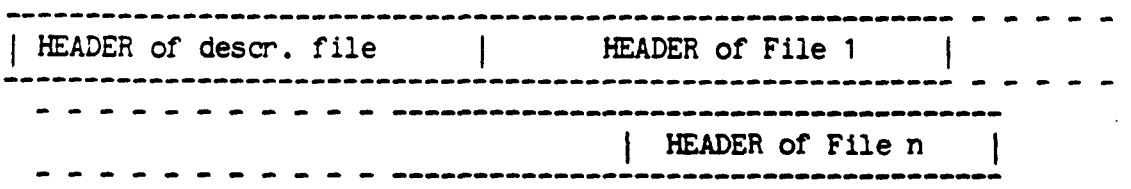
4.3 FILE CODING.

The file coding scheme is a TLV encoding and uses the same principles as those described in section 4.1.1 for TDU's parameter field encoding.

4.3.1 File structure coding.

Each file has the following structure:

The description file has the following structure

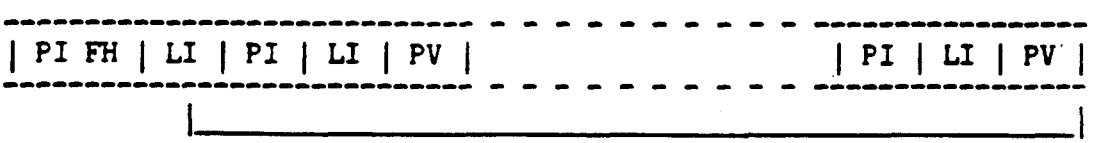


The structure of headers is given in section 4.3.2

4.3.2 File header coding.

The header is a sequence of attributes, each attribute is a group of PI LI PV.

The general structure of a header is the following:



The File Header PI (FH) is used to delimit the header of the file , its associated LI indicates the length of the list of attributes contained in the header. This parameter is always present but if there is no header then LI is set to 0/0.

The File Header PI is coded 3/0 .

The following file attributes are specified:

TABLE - CODING OF FILE ATTRIBUTES

Attribute	PI
File type	2/0
Execution order	2/1
Transfer name	2/2
File name	2/3
Date	2/4
File length	2/5
Destination code	2/6
File coding	2/7
Destination name	2/8
Cost	2/9
User field	2/10
Load address	2/11
Execute address (absolute)	2/12
Execute address (relative)	2/13
Compression mode	2/14
Device	2/15

4.3.2.1 File type

PI File type = 2/0
LI = 0/1
PV = 1 byte coded as follows

4/0 description file
4/1 software file
4/2 data file
4/3 command file
4/4 text file (default value)
All other values are reserved for future extension

4.3.2.2 Execution order

PI Execution order = 2/1
LI = 0/1
PV = 1 byte coded as follows

2/0 don't care (default value)
2/1 highest execution order
7/15 immediate processing

4.3.2.3 Transfer name

PI Transfer name = 2/2
LI = n < 255
PV = n bytes string

In the symmetrical service, the value is made up of keywords.

4.3.2.4 File name

PI File name	= 2/3
LI	= n < 255
PV	= n bytes string

The value is a variable length string containing the file name.

4.3.2.5 Date

PI Date	= 2/4
LI	= n < 13
PV	= n bytes string

The value is a variable length string of an even number of digits, coded in the range 3/0 to 3/9, specifying the date and time in the format yymmddhhmmss. The string may be truncated from the right to give a lower degree of accuracy.

4.3.2.6 File length

PI File length	= 2/5
LI	= n < 9
PV	= n bytes string

The value is a variable length parameter in which the expected length in bytes is coded in absolute binary form using all eight bits of each byte. If the parameter is coded in more than one byte, then the first byte to be transmitted contains the most significant bits.

Note 4.4: If a compression algorithm is used (see 4.3.2.15) the value indicates the size in bytes of the file content before performing the compression algorithm.

4.3.2.7 Destination code

PI Destination code	= 2/6
LI	= 0/1
PV	= 1 byte coded as follows

4/0 do not care (default value)
4/1 foreground memory
4/2 background memory
4/3 cassette tape

All other values are reserved for future extension

4.3.2.8 File coding

PI File coding	= 2/7
LI	= n < 255
PV	= n bytes string

The first byte of the string is coded as follows:

software file :	2/0	text source
	2/1	source in tokenised form
	2/2	intermediate code
	2/3	object code
	2/4	execution code (default value)
data file :	3/0	binary code (default value)
	3/1	codes 2/0 to 7/14 of the videotex primary graphic set and the basic control codes specified in 4.3.4.3.1
command file :	4/0	machine dependant (default value)
	4/1	standard code
Text file :	5/0	other code
	5/1	As for 3/1 above (default value)
	5/2	Videotex code profile 2
	5/3	Videotex code profile 1
	5/4	Videotex code profile 3
	5/5	geometric
	5/6	photographic
	5/7	sound
	All other values are reserved for future extension	

The default values indicated above depend on the value of the parameter file type and if file type is absent the default value is 5/1.

Subsequent bytes, if present, identify by means of a text string a language (such as 'BASIC', 'P-CODE'), or a target processor (such as '6502'). This may optionally be followed by a 2/15 ('/') and an identification of the language dialect (such as '/MSDOS').

4.3.2.9 Destination name

PI Destination name	**= 2/8**
LI	**= n < 255**
PV	**= n bytes string**

4.3.2.10 Cost

PI Cost	**= 2/9**
LI	**= n < 32**
PV	**= n bytes string**

The string must contain displayable characters (codes 2/0 to 7/14 of the videotex primary graphic set).

4.3.2.11 User field

PI User field .	= 2/10
LI	= n < 255
PV	= n bytes string

The string must contain displayable characters (codes 2/0 to 7/14 of the videotex primary graphic set).

4.3.2.12 Load address

PI Load address	= 2/11
LI	= n < 9
PV	= n bytes string

The value is a variable length parameter in which the load address is coded in absolute binary form using all eight bits of each byte. If the parameter is coded in more than one byte, then the first byte to be transmitted contains the most significant bits.

4.3.2.13 Execute address (absolute)

PI Execute address (absolute)	= 2/12
LI	= n < 9
PV	= n bytes string

The value is a variable length parameter in which the absolute execute address is coded in absolute binary form using all eight bits of each byte. If the parameter is coded in more than one byte, then the first byte to be transmitted contains the most significant bits.

4.3.2.14 Execute address (relative)

PI Execute address (relative)	= 2/13
LI	= n < 9
PV	= n bytes string

The value is a variable length parameter in which the relative execute address is coded in absolute binary form using all eight bits of each byte. If the parameter is coded in more than one byte, then the first byte to be transmitted contains the most significant bits.

4.3.2.15 Compression mode

PI Compression mode	= 2/14
LI	= 0/1
PV	= 4/0

The value identifies the compression algorithm as defined below. All other values are reserved for future use. Absence of this parameter means that no compression algorithm is in use.

The use of compression algorithm applies to the file content.
To repeat a byte the following characters should be transmitted:
X, RPT, N.
Where X is the byte to be repeated in the range 0/0 to 15/15,
RPT is the character 1/2,
N is coded 2/0 + n, n being the number of repetitions in the range 0/1 to 5/15.

When this compression algorithm is in use the RPT code (1/2) should be transmitted twice.

4.3.2.16 Device

PI Device	= 2/15
LI	= n < 255
PV	= n bytes string

The value is a variable length string comprising one or more fields. Fields are separated by the character 4/0. The first field should contain a name which is sufficient to distinguish the device type; the use of other fields is unstandardised. The use of the initial character 2/1 ('!') in each field is reserved for standardised devices.

If the first field commences with 2/1,5/0 ('!P'), then the required device is a printer. The coding of the data field is specified in 4.3.4.3.

A third byte may be present in the first field to indicate the minimum printer character repertoire required, as follows:

2/1: codes 2/0 - 2/2, 2/5 - 5/10 and 6/1 - 7/10 of the Videotex primary graphic set, and the control codes specified in 4.3.4.3.1.
Codes 2/3, 2/4, 5/11-5/15, 7/11-7/15 may also be used but the representation of these characters is not guaranted in international communications and may be replaced by national application oriented variants.

2/2: the codes specified for 2/1, plus the mosaic characters in the first supplementary mosaic set

2/3: the codes specified for 2/1, plus the SS2 control code, plus the following G2 characters:2/3, 2/4, 2/6, 2/12, 2/13, 2/14, 2/15, 3/0, 3/1, 4/1, 4/2, 4/3, 4/8, 4/11, 6/10, 7/10.
The following characters must be displayed: à é ù è â ê î ô û ⁄e ¨ oe ç Â È É Î Ô Û OE Ç .

2/4: the codes specified for 2/3, plus SI and SO control codes, plus the characters of the second supplementary mosaic set (G1) except the codes 4/0 to 5/14.

2/5: the alphanumeric repertoire specified in section 2.1.1 of ETS T/TE 06-01 Part 1.

2 the alphanumeric and mosaic repertoires specified in sections 2.1.1 and 2.1.2 of ETS T/TE 06-01 Part 1.

2/7: the codes specified for 4/4, plus DRCS capability.

4/1: the codes specified for 2/1 plus the control codes specified in 4.3.4.3.2

4/2: the codes specified for 2/2 plus the control codes specified in 4.3.4.3.2

4/3: the codes specified for 2/3 plus the control codes specified in 4.3.4.3.2

4/4: the codes specified for 2/4 plus the control codes specified in 4.3.4.3.2

4/5: the codes specified for 2/5 plus the control codes specified in 4.3.4.3.2

4/6: the codes specified for 2/6 plus the control codes specified in 4.3.4.3.2

4/7: the codes specified for 2/7 plus the control codes specified in 4.3.4.3.2

4/15: private use.

All other values are reserved for future extension.

There may follow a string of arbitrary length comprising a sequence of codes of additional control and/or graphic characters or characters sequences which are required to be reproduced (for example, national currency signs, accented characters, selected line drawing characters, graphic renditions).

If the repertoire specification byte is 4/15, the subsequent characters may be used to define the repertoire, and possibly also the page format or special stationery to be used.

The second field, if present and of non-zero length, specifies the page format required. It may be either:

- a decimal number coded using the bytes 3/0 to 3/9, indicating that the printer must be able to print rows of at least this number of characters,

- a character, indicating that each record is to be formatted to a locally specified line length, line breaks being inserted in place of spaces, or following hyphens ('-', 2/13). Permitted characters are 4/12 ('L') indicating that the text is to be left justified, and 4/10 ('J') indicating that text preceding an inserted line-break should be justified to span the available line-length.

4.3.3 Coding of the file content.

4.3.3.1 Description file.

The content of the description file is made of the file headers of all the other files part of the application to upload or to download.

Each file header is introduced by the parameter FH followed by the length indicator. The parameter value contains at least the Transfer name and the Size of the file.

4.3.3.2 Other file.

The other files contain their own data. This data follows the file header. They are not introduced by a parameter identifier.

4.3.4 Content of some specific files.

4.3.4.1 Text file resulting of a T-DIRECTORY Request.

This file contains a list of file Transfer names.

Each Transfer name is identified by one or more key words (maximum 8).
If there is more than one keyword they are separated by "/". A private field may follow the Transfer name, in this case it is introduced by the code 2/0 and it contains displayable characters (coded between 2/1 and 7/14). The maximum length of the file name including separators is of no more than 70 bytes, and the private field length including introducer is of no more than 10 bytes.

Whatever the coding (character coding or videotex coding) each field, including Transfer name and private field, is terminated by CR,LF (0/13, 0/10).

If the coding is videotex, there are no sequence US x,y inside a field and no codes HT, VT, BS. The file is made of pages of 24 rows and 40 characters. A given field does not overlap on two consecutive pages.

4.3.4.2 File of group B and C.

These files are text files. They may be coded in character code or in videotex code.

If the coding is videotex, all the alphamosaic videotex coding can be used. The file is made of pages of 24 rows and 40 characters.

4.3.4.3 Coding of data intended for a standardised printer

This section specifies the coding of data intended for the standardised printer (device !P) in the mass transfer phase.

In the auxiliary device application each transfer is intended to commence a new page or form, although the transfer may extend to more than one page or form of text.

Graphic character codes will be encoded according to the ETSI videotex display syntax and to the device parameter of the file header.

Control codes will be interpreted by the terminal as described in sections 4.3.4.3.1 and 4.3.4.3.2.

4.3.4.3.1 Basic control codes

4.3.4.3.1.1 NULL.

Coded 0/0 . This character should be ignored.

4.3.4.3.1.2 Backspace. BS.

Coded 0/8 . The preferred action is to overprint character(s) previously received; the default action is to print only the second character received.

4.3.4.3.1.3 Line feed. LF.

Coded 0/10.

4.3.4.3.1.4 Clear screen. CS.

Coded 0/12 and with the meaning 'new page'.

4.3.4.3.1.5 Carriage return. CR.

Coded 0/13.

4.3.4.3.2 Extended control codes

The control codes specified for the virtual standardised printer are taken in general from the ETSI videotex display syntax or from ISO 6429. Terminals are expected to recognise the following control code syntaxes:

ESC (2/x ...) 3/x
ESC (2/x ...) 4/x
ESC (2/x ...) 5/x
ESC (2/x ...) 6/x
ESC (2/x ...) 7/x

CSI (3/x ...) (2/x ...) 4/x
CSI (3/x ...) (2/x ...) 5/x
CSI (3/x ...) (2/x ...) 6/x
CSI (3/x ...) (2/x ...) 7/x

Where (2/x ...) denotes zero, one or more occurences of a code 2/x. ESC = 1/11, CSI = 1/11 5/11 or CSI = 9/11.

Any other single byte from columns 0, 1, 8, 9.

Note 4.5: The codes sequences ESC 4/x and ESC 5/x have the same meaning as the codes 8/x and 9/x respectively.

Note 4.6: Control codes and code sequences are non-spacing except for spacing Videotex display attributes (section 4.3.4.3.2.8).

Terminals are expected to interpret and implement the following control codes:

4.3.4.3.2.1 Repeat. RPT.

Coded 1/2, X and with the meaning "repeat the last graphic character a further n times", where n is the binary value of the six least significants bits of X.

4.3.4.3.2.2 Active position address. APA.

Coded 1/15, X', X, Y', Y (all parameters taken from columns 4 to 7 of the code table). This sequence causes the active position to be moved to the row specified by the bytes X' and X,and the column specified by the bytes Y' and Y, provided that the specified row number is equal or higher than the current row number on the page. (The effect of attempting to overwrite text on the current row, or of addressing a column greater than that available on the printer, is not specified). The row and column are specified as 12-bit values using the least significant 6 bits of 2 bytes, the first byte carrying the most significant bits. If both row and column address 0 are specified, then the active position should remain in the row and column in which it was situated before the APA sequence was received.

The operation of the following codes may be implementation dependant:

4.3.4.3.2.3 Unit separators. US.

Coded 1/15, X, Y (X code taken from columns 2 and 3 of the code table). This code sequence identifies the start of non-alphamosaic-coded data which may be ignored by the terminal unless it is within the specified repertoire of the terminal (e.g. DRCS definitions as required by the repertoire specification byte 4/5). The end of the data string which may be ignored will be marked by a US or APA sequence

4.3.4.3.2.4 Partial line down. PLD .

Coded 9/11, 4/11. The following characters within the same record, or until the next CS, US or PLU character, are intended to be printed as subscripts; if PLU is already in effect then PLD has the effect only of cancelling the PLU.

4.3.4.3.2.5 Partial line up. PLU.

Coded 9/11, 4/12. The following characters within the same record ,or until CS, US or PLD character,are intended to be printed as superscripts; if PLD is already in effect then PLU has the effect only of cancelling the PLD.

4.3.4.3.2.6 Select graphic rendition. SGR .

Coded 9/11, (P1, (3/11, P2, (3/11, P3, (3/11, P4,)))) 6/13 where each parameter ,Pn ,may take one of the values:

3/1	bold or increased intensity
3/3	italic
3/4	underlined
3/9	crossed-out

Each parameter affects the representation of subsequent text up to the next SGR sequence, or to the end of the transfer if no further SGR is encountered.

4.3.4.3.2.7 Graphic size modification GSM.

Coded 9/11, P1, 3/11, P2, 2/0, 4/2 where the parameters P1 and P2 are each encoded as decimal numbers using the bytes 3/0 to 3/9, specifying the required height and width respectively of subsequent characters as percentages of the default height and width . Each parameter affects the representation of subsequent text up to the next GSM sequence, or to the end of the transfer if no further GSM is encountered.

4.3.4.3.2.8 Videotex display attributes.

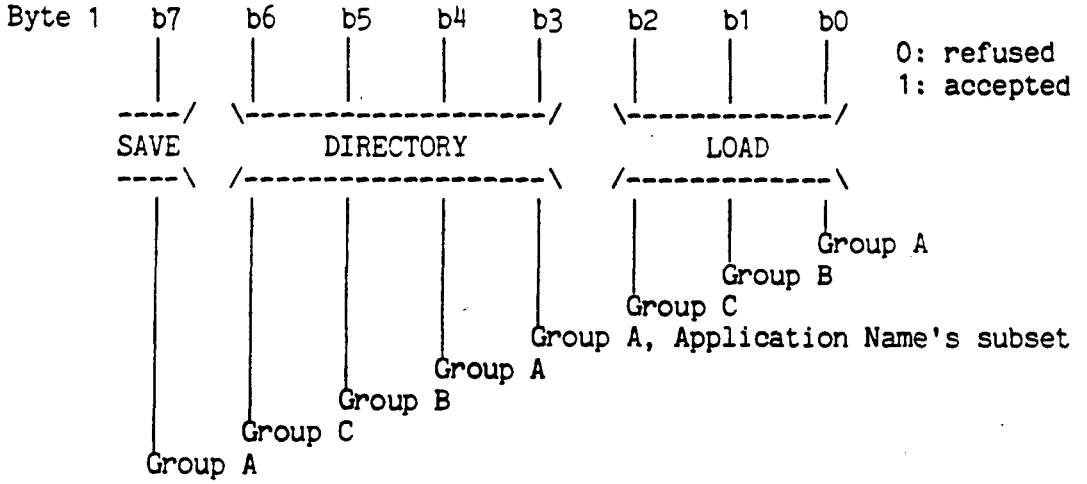
Coded 8/x or 9/x (or alternatively ESC 4/x or ESC 5/x). These attributes may be used to indicate a preferred foreground or background colour or character size for the printed text. (Use of a videotex display attribute to change character size is to be regarded as an alternative to the use of GSM, it does not alter the default height referenced by the GSM command. For example, the code 8/13 (double height) has an identical effect in the terminal to the code sequence 9/11, 3/2, 3/0, 3/0, 3/11, 3/1, 3/0, 3/0, 2/0, 4/2.

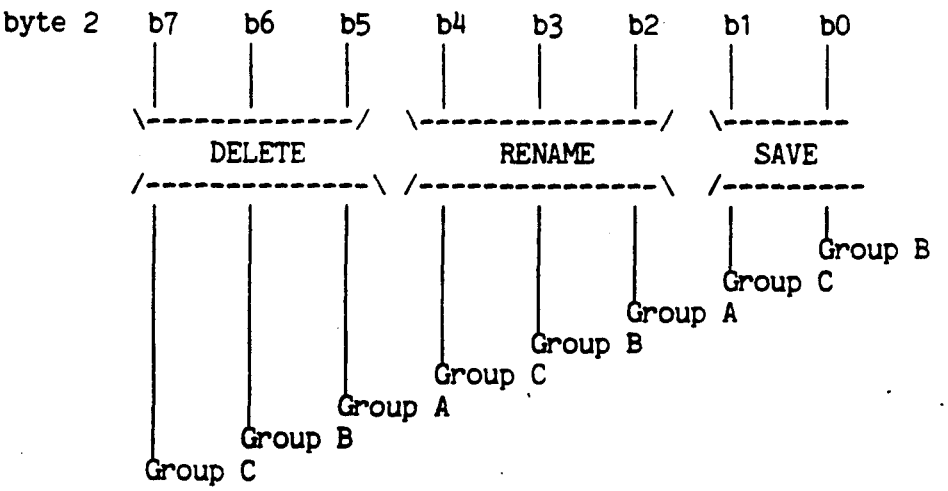
4.4 CODING OF PARAMETERS FOR THE TELESOFTWARE AND PRINTER DEVICE APPLICATIONS.

4.4.1 User data in T-Access and in T-(Access) Response positive.

This parameter is used by the Slave to indicate in which groups the TDUs may be received:

Slave only:





Default value: The primitives T-DIRECTORY, T-LOAD, T-SAVE, T-RENAME, T-DELETE, if they are authorized, are only used in group A, which is equivalent to byte 1 equal to 8/9 and byte 2 equal to 2/4.

4.4.2 User data in T-Directory.

This parameter is coded on 1 byte. It indicates:

3/0 : group A , description file subset,
3/2 : group.B,
3/3 : group C.

Default value: Group A.

4.4.3 Designation in T-Directory.

The Designation parameter is made up of one or more elementary words separated by the codes 2/11 (displayed "+"), 2/15 (displayed "/"), 2/8 (displayed "(") and 2/9 (displayed ")").

In a byte sequence forming a correct sequence (see 2.7 of the symmetrical service Telesoftware and printer device applications),

2/11 will be interpreted as the logical operator OR,
2/15 will be interpreted as the logical operator AND.

Parenthesis are used to indicate the scope of these logical operators.

The maximum number of elementary words is 8.

An elementary word is a byte sequence of no more than 12 bytes. Each byte can take any value between 2/1 and 7/14 excepted 2/8, 2/9, 2/11 and 2/15, moreover this word cannot contain more than one byte 2/10 (displayed "*").

4.4.4 Designation in T-Load and T-Save.

The Designation parameter contains the Transfer name.

A Transfer name is made up of one or more key words. If a Transfer name is made up of several key words, then they are separated by the code 2/15 (displayed "/").

The maximum number of key words in a Transfer name is 8. The maximum length of a Transfer name is 70 bytes including all separators.

A key word is a byte sequence of no more than 12 bytes. Each byte can take any value between 2/1 and 7/14 excepted 2/8, 2/9, 2/10, 2/11 and 2/15.

4.4.5 User data in T-Load, T-Save, T-Rename, T-Delete.

This parameter is coded on 1 byte. It indicates:

3/2 : group B,
3/3 : group C.

Default value: Group A.

5 D-PROTOCOL SPECIFICATION

5.1 DEFINITIONS

DDU : Dialogue Data Unit.

VPDE : Videotex Presentation Data Element (see ETS T/TE 06-01).

Videotex Frame : The data that are retrieved by a single command from a videotex terminal.

BCS : Block Check Sequence.

ED : Error Detection Mechanism.

Translation modes: 4 translation modes are defined in order to provide for 8-bit transparency according to the videotex environment in which the protocol is to be used, these modes are specified in section 5.5.

Mode 1: No translation.
Mode 2: 3-in-4 coding.
Mode 3: Shift scheme 8-bits.
Mode 4: Shift scheme 7-bits.

DDU modes: 7 DDU modes are defined in order to adapt the DDU delimitation codes to the videotex environment in which the protocol is to be used. These modes are specified in section 5.5.

Mode A : 1 byte response encoding
Mode B : 1 byte plus terminator response encoding
Mode C : Symmetrical encoding with a one byte terminator
Mode D : Redefinable response encoding
Mode E : Redifinable response encoding with a End Delimiter
Mode F : Symmetrical encoding with a End Delimiter
Mode G : Symmetrical encoding

Other modes might be added to cover possible identified requirements related to the studies on the symmetrical service.

5.2 GENERAL OVERVIEW OF THE PROTOCOL

5.2.1 Structure of DDUs

The Dialogue Data Units are used to delimit TDU's. DDU's are introduced by the Videotex processable data delimiter and are delimited by their internal syntax (see section 5.5). Optional 8-bit transparency capabilities (translation modes) and optional error detection facilities (BCS) are provided, but either or both of these features may be omitted if they are already provided by other means.

5.2.2 Flags

Flags are associated with D-Set-mode and D-Data DDUs. These flags are used by the error recovery mechanism. Only one of these flags may be set in a given DDU.

The more flag indicates, if set that this is not the last processable data VPDE in the videotex frame, further processable data VPDEs should be expected without request.

The poll flag indicates, if set, that no further processable data VPDEs will be sent until requested, and that the receptor is expected to transmit a D-response-positive if this block and all previous blocks have been received correctly

The confirmation flag indicates, if set that no further processable data VPDEs will be sent until requested, and that the Slave is expected to return the TDU acknowledgement, after this block and all previous blocks have been received correctly and processed by the application. This flag is set according to the semantic of the TDU which is conveyed by the corresponding D-Set mode or D-Data.

In symmetrical service:
The no flag is used when ED is not used and if the D-Data DDU contains only a T-Abort, a T-Typed data, a T-Transfer reject, a T-Read-restart, a T-P-Exception or a T-Response.

The poll flag is used only if ED is in used.

5.2.3 Error detection and recovery

This section gives an overview of the error detection and recovery mechanism.

The error detection is provided by using a sequence numbering or both a sequence numbering and a checksum . The use of error detection can be selected by sending a D-Set mode indicating the use of either sequence numbering (mode E and F only) or both sequence numbering and checksum (all modes).

The main principles of the error recovery mechanism are described below. The complete specification is given in section 5.4. This mechanism makes use of the sequence numbering and the BCS.

On receipt of an erroneous DDU (e.g.detected with an incorrect BCS or DDU out of sequence) the receiver must send a D-Response-negative.

If ED is in use, the TDU's contained in the D-Data DDU's are delivered to the upper layer when a poll or confirmation flag is contained in the last correctly received D-Data.

If ED is not in use the TDU's contained in the D-Data DDU's are delivered as soon as DDU's are received.

When a D-Response-negative is received the sender must send again the DDUs following the last DDU with a poll or confirmation flag set and for which a D-Response-positive or a T-response has been received.

D-Response-negative is only used when ED is in use.

Two timers are defined for the purpose of error detection:

General receive inactivity timer: This timer is only active while processable-data reception is active. It is started by the reception of a processable-data delimiter and restarted by the reception of any data. It is stopped by the reception of a complete DDU, unless the More flag has been set.

DDU-request timer: This timer is started by the transmission by the receptor of a D-Response-positive or a D-Response-negative. It is stopped by the reception of a DDU with a sequence code indicating either the required sequence number or an unnumbered DDU.

5.3 REPERTOIRE AND USE OF DIALOGUE DATA UNITS

This section describes the DDUs and their parameters (see Table 1).

TABLE 1

DIALOGUE DATA UNITS

DDU	Parameters	Mandatory/opt.
D-Set-mode	Translation mode	M
	DDU mode	M
	Flags	M
	Use of sequence numbering	O
	Use of BCS	O
	Define D-Response Positive	O
	Define D-Response Negative	O
	Inactivity timer	O
	Request timer	O
D-Data	Sequence Code	O
	Translation mode	M
	Flags	M
	Reset	O
	Define D-Response Positive	O
	Define D-Response Negative	O
D-U-Abort	None	-
D-Response-positive	None	-
D-Response-negative	None	-

5.3.1 D-set-mode

D-Set-mode announces the intended use of processable data VPDE's. D-Set-mode selects the translation mode and the DDU mode. D-Set-mode resets the sequence number to 0 for both direction of transmission and resets all DDU variables to default unless specified by one of its optional parameters.

5.3.1.1 Content of D-set-mode DDU

3.3.1.1.1 Translation mode.

This parameter defines the 7-8 bits translation technique to be used. The values of this parameter specify that the processable data facility is to be used and define the data coding scheme in use as described in section 5.5. This parameter takes effect immediately,

and will therefore affect the decoding of any subsequent variable-length parameter values in the same VPDE as well as subsequent VPDE's. Translation mode 1 may not be used when DDU mode C is used.

5.3.1.1.2 DDU-mode

This parameter specifies the DDU mode to be used as described in section 5.5. This parameter takes effect immediately.

5.3.1.1.3 Flags

This parameter indicates which flags are associated with D-Set-mode (see section 5.2.2).

5.3.1.1.4 Use of ED

This parameter indicates that either sequence numbering or both BCS and sequence numbering will be used.

5.3.1.1.5 Define D-Response-positive

This parameter may only be used when DDU mode D or E or F are selected,it specifies the coding of D-Response-positive. The redefinition has immediate effect. The default value is 3/0.

5.3.1.1.6 Define D-Response-negative

This parameter may only be used when DDU mode D or E or F are selected, it specifies the coding of D-Response-negative. The redefinition takes effect after the reception of a D-Response-positive. The default value is 3/1.

Note 5.1: To guard against possible deadlock in the event of double error, the sender should recognise both the previous and new response negative until a response positive has been received.

5.3.1.2 Sending D-set-mode

In the basic kernel, D-Set-mode may only be sent by the host. In the symmetrical service D-Set-mode may be sent by both entities.
D-Set-mode must not be preceded by any other processable data VPDE in a videotex frame.

If the poll flag is set, the sender must wait until a D-response-positive or a D-response negative is received.

If the confirmation flag is set the sender must wait until the reception of a TDU.

5.3.1.3 Receiving D-set-mode

On reception of D-Set-mode, the processable data facility is activated and the translation mode and DDU modes are selected according to the parameters of D-set-mode.

If the DDU is accepted and if the poll flag is set, a D-response-positive must be returned.

If the confirmation flag is set the DDU layer waits for a local T-response or an incoming DDU.

If D-set mode or one of its parameters is rejected, a D-U-Abort must be returned.

5.3.2 D-Data

This DDU is used to carry data for the upper layer (TDU's).

5.3.2.1 Content of D-Data DDU

5.3.2.1.1 Sequence code

This parameter contains a sequence number if the sequence numbering is in use (see section 5.4.2) otherwise this parameter is absent.

5.3.2.1.2 Translation mode

As specified in 5.3.1.1.1 above.

5.3.2.1.3 Flags

This parameter indicates which flags are associated with D-Data (see section 5.2.2).

5.3.2.1.4 Reset

This optional parameter enables selective reset of the sequence number to 0, and/or of the code(s) for D-Response-positive and D-Response-negative to default. Default is no reset. In case of conflict with the redefinition parameters (sections 5.3.2.1.5 and 5.3.2.1.6) the most recently received parameter takes precedence. In the case of reset of a D-Response-negative, the reset takes effect after a D-Data with a poll or confirmation flag set has been accepted (see 5.3.1.1.6).

5.3.2.1.5 D-Response-positive

As specified in 5.3.1.1.5 above but default is no change.

5.3.2.1.6 D-Response-negative

As specified in 5.3.1.1.6 above but default is no change.

5.3.2.2 Sending D-Data

In the basic kernel, D-Data may only be sent by the host. In the symmetrical service D-Data may be sent by both entities.
If the poll flag is set the sender must wait until the reception of a D-response positive or negative.
If the confirmation flag is set the sender must wait until the reception of a TDU.

5.3.2.3 Receiving D-Data

On reception of D-Data, the data is delivered to the upper layer if the poll or confirmation flag is set. If the more flag is set the data is buffered.

On reception of a valid D-Data, if the poll flag is set a D-response positive must be sent, if the confirmation flag is set the DDU layer waits for a local T-response or an incoming DDU.

If ED is in use and the D-data is invalid (see 5.4.6) a D-response negative must be sent.

5.3.3 D-U-Abort

This DDU is used to abruptly terminate the use of processable data VPDE.

5.3.3.1 Content of D-U-Abort DDU

This DDU contains no parameter.

5.3.3.2 Sending D-U-Abort

Sending D-U-Abort causes the immediate termination of the DDU layer.

D-U-Abort may be sent at any time by the master after having sent a D-set-mode. No other processable VPDE must be sent if a D-Set mode is sent.
D-U-Abort may be sent at any time by the slave after having received a D-set mode. All other processable VPDEs will be discarded until a D-Set mode is received.

5.3.3.3 Receiving D-U-Abort

Receiving a D-U-Abort causes the immediate termination of the DDU layer. All other processable VPDE will be discarded until a D-set mode is transmitted.

5.3.4 D-Response-positive

D-Response-positive is used to acknowledge a received DDU with the poll flag set.

5.3.4.1 Content of D-Response-positive DDU

No parameter is associated with this DDU.

5.3.4.2 Sending D-Response-positive

A D-Response-positive is sent on the correct reception of a DDU with the poll flag set.

When sending a D-response-positive the DDU-request timer is started.

5.3.4.3 Receiving D-Response-positive

After reception of a D-response positive another DDU may be sent.

5.3.5 D-Response-negative

D-Response-negative is used to indicate an error in the process of receiving DDUs.

5.3.5.1 Content of D-Response-negative DDU

No parameter is associated with this DDU.

5.3.5.2 Sending D-Response-negative

A D-Response-negative may be sent on detection of a DDU error or timer expiration (see section 5.4). When sending a D-response-negative the DDU-request timer is started.

5.3.5.3 Receiving D-Response-negative

On reception of D-response negative, the transmission is resumed from the DDU following the last acknowledged DDU (see section 5.4.6).

5.4 ERROR DETECTION AND RECOVERY MECHANISM

5.4.1 Use of BCS

A checksum (BCS) is associated with each DDU (D-Set mode ,D-Data) if the "Use of BCS" parameter is set in the initial D-Set mode DDU. It is recommended not to use BCS when a data link layer is available.

The BCS is used to detect DDU transmission errors. The BCS encoding is specified in section 5.5.

5.4.2 Use of sequence numbering

Sequence numbering of the DDU (D-Set mode , D-Data) is used if the "Use of BCS" or the "Use of Sequence Numbering" is set in the D-Set mode DDU.

The sequence number is set to 0 when sending or receiving the D-Set-mode DDU. This sequence number is incremented by one with each D-Data.

The first D-Data after D-Set-mode has a sequence number of 1. The sequence number is reset to 0 by using the "reset" parameter in D-Data.

An internal variable Sn contains the value of the next sequence number.
An other internal variable Sa contains the value of the sequence number associated with the last acknowledged D-Data.

Sn and Sa are set to 0 when sending (resp. receiving) D-Set-mode or D-Data with the "reset" parameter.

Sn is incremented by one after having sent a D-Data without the "reset" parameter. If ED is in use, the "sequence code" parameter contains the current value of Sn plus one. When Sn is equal to 31 the incrementation of Sn sets Sn to 0.

Sa is set to Sn when receiving (resp. sending) a D-Response-positive or a T-Response (which is considered as a positive response by the DDU layer).

Note 5.2: When Symmetrical DDU modes are used (C, F, G) the sequence numbering mechanism applies to both directions independently (i.e. a set of Sn and Sa is associated with each direction of transmission). D-Set-mode resets both sets of variables while the "reset" parameter of D-Data resets only the set of variables associated with the direction of transmission of D-Data.

A sequence number is detected as invalid when ED use has been selected and:

- no recovery has been initiated and the sequence code indicates a sequence number different from Sn+1,

- or recovery has been initiated and the sequence code indicates a sequence number greater than Sn or lower than Sa+1,

Note 5.3: all these value comparisons are modulo 32.

5.4.3 Use of flags

If ED is in use, data (i.e. TDU's) are delivered to the upper layer upon reception of a valid DDU with a poll or confirmation flag set or upon reception of a valid D-U-Abort.

Note 5.4: valid DDU's are DDU's which do not lead to detection of DDU exception or error (see 5.4.6 and 5.4.7).

If BCS is not in use, data are delivered to the upper layer as soon as DDU's are received.

Due to limitations of sequence number encoding, the maximum number of subsequent D-Data without poll or confirmation flag must not exceed 31.

5.4.4 Size of DDUs

A parameter of D-Set-mode indicates whether the maximum size of the data contained in D-Data must not exceed 2048 bytes or if there is no limit on this size

If the basic kernel is used or if ED is in use this parameter must indicate that this size is limited to 2048 bytes.

When ED is in used, for buffering reasons, the maximum size of data between two poll or confirmation flags should not exceed 2048 octets.

Note 5.5: All these sizes are the number of bytes which are transmitted on the line taking into account the translation mode mechanism.

5.4.5 Use of Timers

Two timers are defined for the purpose of DDU loss detection:

General receive inactivity timer: This timer is only active while processable-data reception is active. It is started by the reception of a processable-data delimiter and restarted by the reception of any data. It is stopped by the reception of a complete DDU, unless the More flag has been set.

DDU-request timer: It is started by the transmission of a D-Response positive or a D-Response-negative if ED is in use. It is stopped by the reception of a DDU with a sequence code indicating the required sequence number if the ED is in use or by reception of a DDU without sequence code.

5.4.6 Actions in the event of DDU errors

In the event of the following DDU reception errors, the entity which detects the error should transmit a D-Response-negative to request retransmission:

- a DDU with an invalid sequence number (see 5.4.2) (only when ED is in use),

- a DDU in excess of permitted length (only when ED is in use),

- a sequence of DDU's without poll or confirmation flags conveying more than 2048 octets (only when ED is in use),

- a BCS error.

The entity which receives a D-Response-negative must send again all the DDU's and their content from the last ackowledged DDU (DDU starting from Sa+1).

5.4.7 Actions in the event of DDU exceptions

The following exceptions should not cause transmission of a D-response negative but are indicated to the upper layer and may cause higher level error recovery to be initiated:

- reception of D-set-mode DDU on an already established DDU connexion.

- unacceptable parameters or parameter values in a received D-Data DDU, (if ED is not in use, the sequence code parameter value is not significant).

- protocol error (syntactically incorrect DDU, or unrecognised DDU identifier, or unexpected DDU, or, if ED is not in use, DDU in excess of permitted length)

- excessive retransmission of D-Response-negative (more than 5 retransmissions).

- unrecoverable error from the data link layer (if such indication is available).

- expiration of general receive inactivity timer or DDU request timer when no sequence numbering is in use.

- a VPDE other than a processable data VPDE is received during the reception of a processable data VPDE (that is a single 1/15 (US) character followed by a character other than 3/14). In this case the subsequent VPDE should be processed.

5.4.8 Actions in the event of timer expiration

If the general receive inactivity timer or the DDU request timer expires,and if the sequence numbering is in use their recovery may be attempted by issuing a D-response negative. The number of retransmision of D-response negative is limited to 5 (see 5.4.7).

5.5 CODING

5.5.1 Translation modes.

The following transcoding schemes apply to all variable length DDU parameter and the whole of each TDU.

5.5.1.1 Mode 1 (No translation)

Under this scheme no translation of data is performed, except that all US (1/15) characters in the processable data are represented by two contiguous US characters in the transmitted data stream.

5.5.1.2 Mode 2 (3-in-4 coding)

Each group of three bytes in the processable data is mapped into four bytes for transmission as shown in Table 4. Any remaining group of one or two bytes at the end of a block of data is mapped into two or three bytes respectively, with undefined bits set to zero. (A block consists of all bytes of a DDU to which the translation algorithm is applied (see 5.5.3.1)).

5.5.1.3 Mode 3 (shift scheme - 8-bits)

In this scheme, bytes of processable data are each mapped into one or two bytes of transmitted data as shown in Table 2; in mode 3 the most significant bit of each transmitted byte is taken into account. Note that most of the conversions are optional. Either entity should be prepared to accept any mixture of converted or unconverted data in these cases.

5.5.1.4 Mode 4 (shift scheme - 7-bits)

In this scheme bytes of processable data are each mapped into one or two bytes of transmitted data as shown in Table 3; in mode 4 the most significant bit of each transmitted byte is not taken into account. Note that some of the conversions are optional. Either entity should be prepared to accept any mixture of converted or unconverted data in these cases.

TABLE 2 - CODE CONVERSION FOR 8 BITS SHIFT SCHEME (MODE 3)

Processable data	Sender's conversion	Optional/Mandatory		Transmitted data
		mode C	A,B,D,E,F,G	
0/0 - 0/12	7/14 x+5/0	-	O	7/14, 5/0 - 5/12
0/13	7/14 x+5/0	M	O	7/14, 5/13
0/14- 01/14	7/14 x+5/0	-	O	7/14, 5/14- 6/14
1/15	7/14 x+5/0	M	M	7/14, 6/15
2/0	7/13	-	O	7/13
2/1 - 7/10	none			2/1 -7/10
7/11	7/11 x-5/8	M	M	7/11, 2/3
7/12,7/13	7/11 x-5/8	-	M	7/11, 2/4 - 2/5
7/14	7/11 x-5/8	M	M	7/11, 2/6
7/15	7/11 x-5/8	-	M	7/11, 2/7
8/0 - 13/0	7/11 x-5/8	-	O	7/11, 2/8 - 7/8
13/1- 15/15	7/14 x+5/0	-	O	7/14, 2/1 - 4/15

- : irrelevant
O : Optional
M : Mandatory

TABLE 3 - CODE CONVERSION FOR 7 BITS SHIFT SCHEME (MODE 4)

Processable data	Sender's conversion	Optional/ Mandatory	Transmitted data
0/0 - 1/14	7/14, x+5/0	O	7/14, 5/0 - 6/14
1/15	7/14, 6/15	M	7/14, 6/15
2/0	7/13	O	7/13
2/1 - 7/10	none	-	2/1 - 7/10
7/11 - 7/15	7/11, x-5/8	M	7/11, 2/3 - 2/7
8/0 - 13/0	7/11, x-5/8	M	7/11, 2/8 - 7/8
13/1 - 15/15	7/14, x+5/0	M	7/14, 2/1 - 4/15

- : irrelevant
O : Optional
M : Mandatory

Note 5.6: In mode 4 the transmitted bytes may have the most significant bit set.

TABLE 4 - 3-IN-4 CODING SCHEME

Processable-data sequence	3 bytes	3 bytes	. .	1,2 or 3 bytes
Transmitted-data sequence	4 bytes	4 bytes	. .	2,3 or 4 bytes respectively

Within each group the bits are mapped as follows, where 'bxy' denotes bit y of byte x in the user data. Bit 7 is not taken into account by this scheme but may be determined by characteristics of the transmission path in use (for example, if parity is required).

a) Three bytes of processable data

Transmitted	b7	b6	b5	b4	b3	b2	b1	b0
1st byte	X	1	b17	b16	b27	b26	b37	b36
2nd byte	X	1	b15	b14	b13	b12	b11	b10
3rd byte	X	1	b25	b24	b23	b22	b21	b20
4th byte	X	1	b35	b34	b33	b32	b31	b30

b) Two bytes of processable data at end of sequence

Transmitted	b7	b6	b5	b4	b3	b2	b1	b0
1st byte	X	1	b17	b16	b27	b26	0	0
2nd byte	X	1	b15	b14	b13	b12	b11	b10
3rd byte	X	1	b25	b24	b23	b22	b21	b20

Note 5.7: in the case of the checksum, b0-b7 of the checksum map to b10-b17 , and b8-b15 of the checksum map to b20-b27 respectively (see 5.5.4).

c) One byte of processable data at end of sequence

Transmitted	b7	b6	b5	b4	b3	b2	b1	b0
1st byte	X	1	b17	b16	0	0	0	0
2rd byte	X	1	b15	b14	b13	b12	b11	b10

5.5.2 DDU MODES.

The DDU's structure depends on the DDU mode :

5.5.2.1 Basic kernel

- Delimitation:

In mode A ,B and D the D-set mode and D-data are delimited using a length indicator (LI).

In mode C the DDU's are delimited with an end delimiter (0/13).

In mode E the DDU's are delimited with an end delimiter (1/15 ...)

- Coding of terminal responses(D-response positive, D-response negative,D-U-abort)

In mode A only the command identifier (CI) is transmitted.

In mode B the CI is transmitted followed by 1/12.

In mode C the CI is transmitted preceeded by 1/15, 3/14 (US '>') and followed by the end delimiter 0/13 . The TDU's sent by the terminal are transmitted in a D-data preceeded by 1/15, 3/14 and followed by 0/13.

In mode D and E the CI of D-U-abort is transmitted, and the string defined in D-set mode or D-data parameter (or its default value) is transmitted for D-response positive or negative.

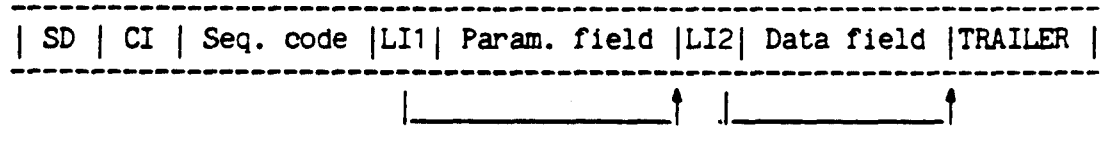
5.5.2.2 Symmetrical service

In the symmetrical service the TDUs are transmitted in D-Set-mode or in D-Data.

5.5.3 Coding of DDU's.

5.5.3.1 Structure of DDU's.

The general structure of DDU's is the following:



SD: Start delimiter coded 1/15, 3/14 (US '>').
This start delimiter is present in all DDUs except in DDUs sent by the terminal in mode A, B, D, E.

CI: Command identifier.
This field is a one byte field coded as described in section 5.5.3.2. (except in mode D, E and F for D-response positive or negative where it is replaced with a redefinable string).

Sequence code:
This field is present in D-set mode and D-data only if ED use is selected.
Its coding is given in section 5.5.3.2.2.1.

LI1: This field indicates the length of the parameter field.
It is coded as described in 5.5.3.1.1 .
In D-Set mode this field is always present .
In D-Data this field is only present in mode D and E.
In other DDU's this field is absent .

Parameter field :
This field is coded as described in 5.5.3.1.2.
This field may only be present when LI1 is present and different from 0.

LI2:This field indicates the length of the data field,it is coded as described in 5.5.3.1.1 .
This field is only present in D-set mode and D-data in modes A,B D, E, F and G.

Data field :
This field may only be present in D-Set-mode and D-Data DDU's.

TRAILER:
In mode A,B,D and G:
This field is only present in D-Set-mode and D-Data if use of BCS has been selected.
Its coding is given section 5.5.4 .

In mode C:
This field contains BCS in the same conditions that in the previous modes.
The trailer is ended in all DDU's in mode C by an End Delimiter.
It is coded 0/13 (Carriage Return).

In mode B:
This field is present in the terminal responses.
It is coded 1/12 ('#') .

In mode E and F:
This field contains either the sequence US,3/14, 2/X followed either by BCS if BCS is in use or the next US sequence.

X contains the flag, in this mode the flag value in the CI parameter is set to "00".

The translation modes specified in 5.5.1 apply to all bytes of D-Set-mode and D-Data except SD, End Delimiter, CI, BCS (see 5.5.4) and Sequence code fields.

5.5.3.1.1 Length Indicator field (LI).

The length indicator is used to indicate the length of a field. The value of the LI field is expressed as a binary number representing the length in octets of the associated parameter field (i.e. the value of the LI field does not include itself).
LI fields indicating lengths within the range 0-254 shall comprise one byte.
LI fields indicating lengths within the range 255-65534 shall comprise three bytes. The first byte shall be coded 15/15 and the second and the third bytes shall contain the length of the associated parameter field with the high order bits in the first of these two bytes.

5.5.3.1.2 Parameter field.

The parameter field comprises zero one or more groups of:

- Parameter Identifier field (PI): A single byte identifying the parameter. The coding of PIs is specified in Table 6.

- Length Indicator field (LI): This length indicator is used to indicate the length of the following parameter value. This LI is coded as specified in 5.5.3.1.1.

- Parameter Value field (PV): The coding of the parameter values is specified in section 5.5.3.2.

The absence of a parameter in a parameter field means that the default value applies if defined.

If a PV field is absent, the whole parameter group must be absent (i.e. LI may not be equal to 0).

In order to ensure easier migration to enhanced versions of the protocol the following rules should apply:

- parameter group containing an undefined parameter identifier should be ignored in reception.

- reserved bits in a parameter value are set to zero by the sender and ignored in reception.

-the parameters may appear in any order in a parameter field .They must not be repeated.

Bits in a byte are numbered from b0 (least significant) to b7 (most significant).

5.5.3.2 DDU's encoding

TABLE 9 - CODING OF DDU COMMAND IDENTIFIERS.

DDU	CI/bcs seq.code	CI/no bcs seq.code	CI/no bcs no/seq.code
D-Set mode	7/x	6/x	4/x
D-Data	5/x	5/x	5/x
D-U abort	3/9	3/9	3/9
D-Response-positive	3/0	3/0	3/0
D-Response-negative	3/1	3/1	3/1

5.5.3.2.1 x values.

bits b1,b0 = 0 0 : translation mode 4
= 0 1 : translation mode 2
= 1 0 : translation mode 3
= 1 1 : translation mode 1

bits b3,b2 = 0 0 :no flag set
= 0 1 :confirmation flag set
= 1 0 :more flag set
= 1 1 :poll flag set

In mode E and F the flags are contained in the trailer. In this case in the CI parameter the bits b3,b2 are set to "00"; in the trailer bits b3,b2 of 2/x are set to "11" and the flag are coded in b1,b0 according to table above.

5.5.3.2.2 parameters encoding.

TABLE 5 - CODING OF DDU PARAMETER IDENTIFIERS.

Parameter	PI
Sequence code	4/0-5/15,6/0
Define D-response-positive	2/1
Define D-response-negative	2/2
DDU mode	2/3
Inactivity timer	2/4
Request timer	2/5
Reset	2/6

Note 5.8: Translation mode, flags are part of the DDU code.

5.5.3.2.3 D-Set-mode.

SD		= 1/15 , 3/14
CI D-Set mode		= 7/x or 6/x or 4/x
*	Sequence code	= 4/0 (see 5.5.3.2.3.1)
	LI1	= 1 byte (see 5.5.3.1.1)
	PI DDU mode	= 2/3
	LI	= 0/1
	PV	= 1 byte (see 5.5.3.2.3.2)
*	PI Define D-Resp-pos.	= 2/1
	LI	= n < 17
	PV	= string (see 5.5.3.2.3.3)
*	PI Define D-Resp-neg	= 2/2
	LI	= n < 17
	PV	= string (see 5.5.3.2.3.4)
*	PI Inactivity timer	= 2/4
	LI	= 0/1
	PV	= 1 byte (see 5.5.3.2.3.5)
*	PI DDU request timer	= 2/5
	LI	= 0/1
	PV	= 1 byte (see 5.5.3.2.3.6)
* LI2		= 1 or 3 bytes (see 5.5.3.1.1)
* Data		= n bytes
* Trailer		
*	BCS	= 3 bytes (see 5.5.4)
*	End delimiter	= 1 byte
or		
*	End delimiter	= 1/15,3/14,2/x (see 5.5.3.1)
*	BCS	= 3 bytes (see 5.5.4)

* : The presence of the field is optional and subject to the conditions indicated in the referred section.

5.5.3.2.3.1 This field is only present if error detection is provided.

5.5.3.2.3.2 DDU mode parameter value.
This parameter is coded on 1 byte. Bits b3,b5,b6,b7 are reserved.

bits b2,b1,b0 = 0 0 0 : mode A
= 0 0 1 : mode B
= 0 1 0 : mode C
= 0 1 1 : mode D
= 1 0 0 : mode E
= 1 0 1 : mode F
= 1 1 0 : mode G

bit b4 = 0 : D-Data size is limited to 2048 bytes,
= 1 : D-Data size is not limited.

5.5.3.2.3.3 Define D-response-positive parameter value.

String of maximum length 16 bytes redefining the D-response-positive DDU. The default value is 3/0.

Note 5.9: It is the responsibility of the Master to ensure that the characters of the string cause no problem in the bearer service.

This parameter is only present in mode D, E and F.

5.5.3.2.3.4 Define D-Response-negative parameter value.

String of maximum lenght 16 bytes redefining the D-response-negative DDU. The default value is 3/1.

Note 5.10: It is the responsibility of the Master to ensure that the characters of the string cause no problem in the bearer service.

This parameter is only present in mode D, E and F.

5.5.3.2.3.5 Inactivity timer parameter value.

This parameter is coded on 1 byte. The timeout interval is coded in binary form in increments of 1 s. The value 0/0 (default value) has the special significance of 'timer disabled'.

5.5.3.2.3.6 DDU Request timer parameter value.

This parameter is coded on 1 byte. The timeout interval is coded in binary form in increments of 1s. The value 0/0 (default value) has the special significance of 'timer disabled'.
This timer is only present if ED is in use.

5.5.3.2.4 D-Data.

	SD	= 1/15,3/14
	CI D-Data	= 5/x
*	Sequence code	= 4/0 - 5/15 , 6/0 (see 5.5.3.2.4.1)
*	LI1	= 1 byte (see 5.5.3.1.1)
*	PI Define D-Resp-pos.	= 2/1
	LI	= n < 17
	PV	= string (see 5.5.3.2.4.2)
*	PI Define D-Resp-neg.	= 2/2
	LI	= n < 17
	PV	= string (see 5.5.3.2.4.3)
*	PI Reset	= 2/6
	LI	= 0/1
	PV	= 1 byte (see 5.5.3.2.4.4)
*	LI2	= 1 or 3 bytes (see 5.5.3.1.1)
*	Data	= n bytes
*	Trailer	
*	BCS	= 3 bytes (see 5.5.4)
*	End delimiter	= 1 byte
	or	
*	End delimiter	= 1/15,3/14,2/x (see 5.5.3.1)
*	BCS	= 3 bytes (see 5.5.4)

* : The presence of the field is optional and subject to the conditions indicated in the referred section.

5.5.3.2.4.1 Sequence code coding

bits b7,b6,b5,b4,b3,b2,b1,b0	
0 1 0 0 0 0 0 0	sequence number modulo 32
0 1 0 0 0 0 0 1	
0 1 0 - - - - -	
0 1 0 - - - - -	
0 1 0 1 1 1 1 1	
0 1 1 0 0 0 0 0	6/0 reset sequence number to 0 (The sequence number of the current DDU is 0 and the next one will be coded 4/1).

This field is only present if error detection is in use.

5.5.3.2.4.2 Define D-response-positive parameter value

As for 5.5.3.2.3.2 above but the default value of this parameter is "no change".

5.5.3.2.4.3 Define D-response-negative

As for 5.5.3.2.3.3 above but the default value of this parameter is "no change".

5.5.3.2.4.4 Reset value.
This parameter is coded on 1 byte. Bits b2,b3,b4,b5,b6,b7 are reserved.

bits b1,b0 = 0 0 : no reset (default value)
= 0 1 : D-Response-positive is reset to default value
= 1 0 : D-Response-negative is reset to default value
= 1 1 : D-Response-positive and D-Response-negative are reset to default values

5.5.3.2.5 D-U-Abort.

CI D-U-Abort = 3/9

5.5.3.2.6 D-Response-positive.

CI D-Response-positive = 3/0

5.5.3.2.7 D-Response-negative.

CI D-Response-negative = 3/1

5.5.4 BCS coding.

The BCS applies to all transmitted bytes of the D-set mode or D-data DDUs from the first byte following the start delimiter up to and including the last byte preceding the BCS field.

The BCS field is coded in 3 bytes using the 3-in-4 coding (see 5.5.1.2).

The BCS is a 16 bits value calculated as follows:

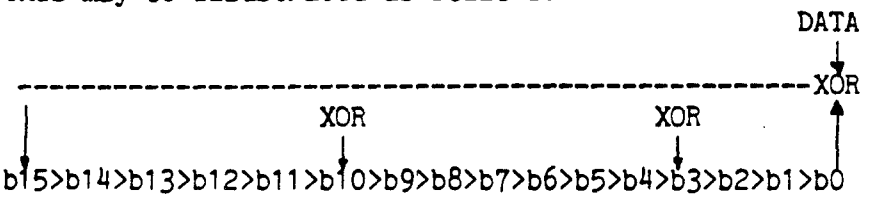
The checksum is the 16-bit frame checking sequence specified in CCITT Recommendation X.25, formed by division by the polynomial:

$$x^{16} + x^{12} + x^5 + 1$$

This may be calculated according to the following procedure.

The checksum, b15 .....b0, is initialised to all 1's. It is then modified in turn for each bit of the data. For this purpose the data is treated as 8-bit bytes, and the least significant bits in each byte are taken first; if the most significant bit of each data byte is a parity bit it is set to zero before the checksum calculation. The modification consists of:
a) an exclusive-OR of b0 with the next data bit,
b) a one-bit shift in which a zero bit is brought into b15 of the checksum.
c) an exclusive-OR of the result of (a) with the new bits b15, b10 and b3 of the checksum.

This may be illustrated as follows:



The checksum is finally complemented (1's complement) and mapped into three bytes as described in section 5.5.1.2 and illustrated in Table 4 (b).

5.5.4.1 Example of CRC calculation

Figure 5 illustrates the calculation of the CRC checksum:

bit 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Next data bit	Next data byte
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	/
1	0	0	0	1	0	1	1	1	1	1	1	0	1	1	1	0	7
1	1	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1	
0	1	1	0	0	0	0	0	1	1	1	1	1	0	0	1	0	
1	0	1	1	0	1	0	0	0	1	1	1	0	1	0	0	0	
0	1	0	1	1	0	1	0	0	0	1	1	1	0	1	0	0	
0	0	1	0	1	1	0	1	0	0	0	1	1	1	0	1	0	
1	0	0	1	0	0	1	0	1	0	0	0	0	1	1	0	0	4
0	1	0	0	1	0	0	1	0	1	0	0	0	0	1	1	0	/
1	0	1	0	0	0	0	0	1	0	1	0	1	0	0	1	0	0
1	1	0	1	0	1	0	0	0	1	0	1	1	1	0	0	0	
0	1	1	0	1	0	1	0	0	0	1	0	1	1	1	0	1	
1	0	1	1	0	0	0	1	0	0	0	1	1	1	1	1	0	
1	1	0	1	1	1	0	0	1	0	0	0	0	1	1	0	0	
1	1	1	0	1	0	1	0	0	1	0	0	1	0	1	1	0	
1	1	1	1	0	0	0	1	0	0	1	0	1	1	0	1	0	4
1	1	1	1	1	1	0	0	1	0	0	1	1	1	1	0	0	/
0	1	1	1	1	1	1	0	0	1	0	0	1	1	1	1	0	0
1	0	1	1	1	0	1	1	0	0	1	0	1	1	1	1	0	
1	1	0	1	1	0	0	1	1	0	0	1	1	1	1	1	1	
0	1	1	0	1	1	0	0	1	1	0	0	1	1	1	1	0	
1	0	1	1	0	0	1	0	0	1	1	0	1	1	1	1	1	
0	1	0	1	1	0	0	1	0	0	1	1	0	1	1	1	1	
0	0	1	0	1	1	0	0	1	0	0	1	1	0	1	1	1	1
0	0	0	1	0	1	1	0	0	1	0	0	1	1	0	1	1	/
0	0	0	0	1	0	1	1	0	0	1	0	0	1	1	0	1	1
1	0	0	0	0	0	0	1	1	0	0	1	1	0	1	1	0	5
1	1	0	0	0	1	0	0	1	1	0	0	0	1	0	1	0	
1	1	1	0	0	1	1	0	0	1	1	0	1	0	1	0	0	
0	1	1	1	0	0	1	1	0	0	1	1	0	1	0	1	0	
1	0	1	1	1	1	0	1	1	0	0	1	0	0	1	0	1	
1	1	0	1	1	0	1	0	1	1	0	0	0	0	0	1	1	3
0	1	1	0	1	1	0	1	0	1	1	0	0	0	0	0	1	/
1	0	1	1	0	0	1	0	1	0	1	1	1	0	0	0	1	1
1	1	0	1	1	1	0	1	0	1	0	1	0	1	0	0	1	4
1	1	1	0	1	0	1	0	1	0	1	0	0	0	1	0	0	
0	1	1	1	0	1	0	1	0	1	0	1	0	0	0	1	0	
1	0	1	1	1	1	1	0	1	0	1	0	0	0	0	0	0	
0	1	0	1	1	1	1	1	0	1	0	1	0	0	0	0	0	
0	0	1	0	1	1	1	1	1	0	1	0	1	0	0	0	0	3
0	0	0	1	0	1	1	1	1	1	0	1	0	1	0	0	0	/
0	0	0	0	1	0	1	1	1	1	1	0	1	0	1	0	1	0
1	0	0	0	0	0	0	1	1	1	1	1	1	1	0	1	1	
0	1	0	0	0	0	0	0	1	1	1	1	1	1	1	0	0	
0	0	1	0	0	0	0	0	0	1	1	1	1	1	1	1	0	
1	0	0	1	0	1	0	0	0	0	1	1	0	1	1	1	Final value	
0	1	1	0	1	0	1	1	1	1	0	0	1	0	0	0	Complement	

These 16 bits map for transmission into:

1st byte: X 1 1 1 0 1 0 0
2nd byte: X 1 0 0 1 0 0 0
3rd byte: X 1 1 0 1 0 1 1

ANNEX A

A Presently used processable data protocol for file transfer from host to terminal.

This annex A is not an integral part of the standard.

VIDEOTEX PROCESSABLE DATA FOR LIMITED APPLICATIONS

CONTENTS

TABLES

FIGURES

0 INTRODUCTION

The Videotex Processable-data facility specified in this Annex is an alternative to the more general Videotex Processable-data protocols described in the main body of this Recommendation. This alternative, which does not form an integral part of the Recommendation, is primarily intended to enable telesoftware downloading from services which may be limited to the storage of pre-defined character sequences accessible only by conventional videotex commands. The facility provides particularly for the downloading of files of data; these files may contain computer software, but other file types are not precluded.

This subset also provides for data to be reliably transferred to devices associated with a videotex terminal under control of the service. In addition to transparent transfer, specific provision is made for passing data to a printer, but other devices may be added later.

It should be noted that terminals requiring only the facility to transfer data to auxiliary devices need not implement the functions specified in sections 4.3 and 6.3. Additionally, for applications requiring the transfer of blocks of data of limited length, it is not necessary to implement the functions of sections 4.2 and 6.2.

In this document the term 'Videotex frame' is used to denote the data that is retrieved from a service by a single command from a terminal (compare CCITT Recommendation F.300).

0.1 Processable data, including telesoftware files, data for printing, file parameters and control data related to the downloading procedure, is transmitted by means of Telesoftware Data Units (TDU's).

0.2 In addition Dialogue Data Units (DDU's) are used to delimit TDU's, to enable terminals to ensure that TDU's are correctly received, and to enable terminals to automatically prompt for the transmission of further videotex frames as required. Optional 8-bit transparency capabilities are provided, and optional error-detection facilities are described in Annex A, but either or both of these features may be omitted if they are already provided by a particular videotex service.

0.3 Coding Structure. Each Processable-data Videotex Presentation Data Element (VPDE) comprises a DDU, which may be followed by one or more TDU's (Figure 1).

DDU's are introduced by the processable-data delimiter, 'US >' (1/15, 3/14). According to the value of the next byte, the DDU may be a D-Command (used to control the decoding of DDU's, and related procedures), a D-Data or a D-End Group DDU. All DDU's are delimited by their internal syntax (see section 7).

D-Data DDU's always introduce a TDU. Certain D-Command DDU's may also introduce a TDU; if they do not then they must be immediately followed by a delimiter.

The first byte in each TDU defines the type of the TDU. This byte is followed by a parameter field coded using a type-length-value (TLV) syntax. According to the type of TDU, the parameter field may be followed by a data field which continues until a new delimiter is encountered, by a further TDU, or may be immediately followed by a delimiter.

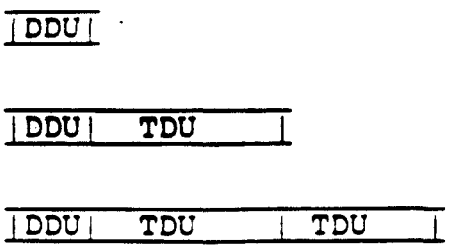
It should be noted that a processable-data VPDE may include a "US US" sequence; this does not terminate the VPDE.

Examples of applications using the syntax defined in this document are described in Annex B.

FIGURE 1 STRUCTURE OF PROCESSABLE DATA VPDE's

a) General Structure

The following VPDE structures may occur:



b) Examples of VPDE Structure

The following VPDE's might form part of a simple telesoftware downloading sequence. Each row comprises one VPDE.

D-Data	T-Associate	T-Capability Spec
*		

D-End Group
*

D-Data	T-Filespec
*	

D-End Group
*

D-Data	T-Write-Start
*	

D-Data	T-Write-End
*	

D-End Group
*

Notes:
* Other VPDE's may be present at these points.

1 MODES OF OPERATION

For the purposes of processable-data, the default mode of operation of the terminal is mode 0, in which the processable-data capabilities are not invoked, and all received data are passed directly to the videotex display.

The processable-data capabilities are invoked by selecting a mode other than 0; according to the mode selected, the following transcoding schemes apply to all variable-length DDU parameters and the whole of each TDU. The transcoding algorithm is to be re-initialised, if necessary, at the beginning of each field to which it is applied.

1.1 Mode 1 (No translation)

Under this scheme no translation of data is performed, except that all US (1/15) characters in the processable data are represented by two contiguous US characters in the transmitted data stream.

1.2 Mode 2 (3-in-4 coding)

Each group of three bytes in the processable data is mapped into four bytes for transmission as shown in Figure 2. Any remaining group of one or two bytes at the end of a block of data is mapped into two or three bytes respectively, with undefined bits set to zero.

1.3 Mode 3 (Shift scheme - 8-bit)

In this scheme bytes of processable data are each mapped into one or two bytes of transmitted data as shown in Table 1; in mode 3 the most-significant-bit of each transmitted byte is taken into account. Note that most of the conversions are optional at the discretion of the service; the terminal should be prepared to accept either converted or unconverted data in these cases.

1.4 Mode 4 (Shift scheme - 7-bit)

In this scheme bytes of processable data are each mapped into one or two bytes of transmitted data as shown in Table 1; in mode 4 the most-significant-bit of each transmitted byte is not taken into account. Note that some of the conversions are optional at the discretion of the service; the terminal should be prepared to accept either converted or unconverted data in these cases.

TABLE 1 - CODE CONVERSIONS FOR SHIFT SCHEMES

Processable data	Service's conversion	Optional/ mandatory 3	4	Transmitted data*
0/0 - 1/14	7/14, x+80 (x+50hex)	O	O	7/14, 5/0 - 6/14
1/15	7/14, 6/15	M	M	7/14, 6/15
2/0	7/13	O	O	7/13
2/1 - 7/10	none			2/1 - 7/10
7/11 - 7/15	7/11, x-88 (x-58hex)	M	M	7/11, 2/3 - 2/7
8/0 - 13/0	7/11, x-88 (x-58hex)	O	M	7/11, 2/8 - 7/8
13/1 - 15/15	7/14, x+80 (x+50hex)	O	M	7/14, 2/1 - 4/15

*In mode 4 the transmitted bytes may have the most significant bit set.

FIGURE 2
3-IN-4 CODING SCHEME

Processable-data sequence	3 bytes	3 bytes	. .	1, 2 or 3 bytes
Transmitted data sequence	4 bytes	4 bytes	. .	2, 3 or 4 bytes respectively

Within each group the bits are mapped as follows, where 'bxy' denotes bit y of byte x in the user data. Bit 7 is not taken into account by this scheme but may be determined by characteristics of the videotex service or transmission path in use (for example, if parity is required).

a) Three bytes of processable data

Transmitted	b7							b0
1st byte	X	1	b17	b16	b27	b26	b37	b36
2nd byte	X	1	b15	b14	b13	b12	b11	b10
3rd byte	X	1	b25	b24	b23	b22	b21	b20
4th byte	X	1	b35	b34	b33	b32	b31	b30

b) Two bytes of processable data at end of sequence

Transmitted	b7							b0
1st byte	X	1	b17	b16	b27	b26	0	0
2nd byte	X	1	b15	b14	b13	b12	b11	b10
3rd byte	X	1	b25	b24	b23	b22	b21	b20

(Note: in the case of the checksum, b0 - b7 of the checksum map to b10 - b17, and b8 - b15 of the checksum map to b20 - b27 respectively (see Annex).)

c) One byte of processable data at end of sequence

Transmitted	b7							b0
1st byte	X	1	b17	b16	0	0	0	0
2nd byte	X	1	b15	b14	b13	b12	b11	b10

2 REPERTOIRE OF DIALOGUE DATA UNITS

This section describes the DDU's and their parameters (see Table 2).

2.1 D-Set mode

D-Set mode announces the intended use of processable-data VPDE's. The same VPDE is also used to terminate their use (set mode 0).

D-Set mode carries a sequence code and must not be preceded by any other processable-data VPDE in a videotex frame. If it is used to set a mode other than mode 0, D-Set mode must be an unnumbered block (see section 7.2).

D-Set mode resets the expected sequence number to 0 and resets all DDU variables to default unless specified by one of the following optional parameters:

TABLE 2
DIALOGUE DATA UNITS

DDU	Parameters	Mandatory /optional
D-Set mode*	Sequence code	M
	Mode	O
	Define D-response positive	O
	Define D-response negative	O
	Define D-response mode-reject	O
	Define D-response token-give	O
	Define general receive inactivity timeout	O
	Define poll timeout	O
D-Control	Sequence code	M
	Mode	O
	Define D-response positive	O
	Define D-response negative	O
	Define D-response token-give	O
	Define general receive inactivity timeout	O
	Define poll timeout	O
	Reset	O
D-Data**	Sequence code	M
D-End group	Flags	M
D-U-Abort***	Sequence code	M

* This DDU may be followed by a TDU unless it sets mode=0.
** This DDU must be followed by a TDU.
*** This DDU may be followed by a TDU.

2.1.1 Mode. This parameter defines the 7-8 bit translation technique to be used. Mode 0 corresponds to termination of the use of processable-data VDPE's and only the D-Set mode VPDE will then be recognised. Other modes specify that the processable-data facility is to be used and define the data coding scheme in use as described in section 1. This parameter takes effect immediately, and will therefore affect the decoding of any subsequent variable-length parameter values in the same VPDE as well as subsequent VPDE's. Default is mode 0 (release).

2.1.2 Define D-response positive. This parameter specifies the code(s) which may be sent by the terminal to acknowledge correctly received frames when requested by the poll flag (see section 2.4), and to acknowledge a D-Set mode or D-Control (mode = 0). The redefinition has immediate effect. Default is '0'.

2.1.3 Define D-response negative. This parameter specifies the code(s) which may be sent by the terminal to request retransmission of incorrectly received data (sequence code and/or BCS error), or to recover from a timeout. The redefinition takes effect after a D-End group with a poll or data token flag set has been accepted. (Note: to guard against possible deadlock in the event of double error, the service should recognise both the previous and new response negatives until a response positive has been received.) Default is '1'.

2.1.4 Define D-response mode-reject. This parameter specifies the code(s) which may be sent by the terminal to reject the D-Set mode DDU. The redefinition has immediate effect. Default is '9'.

2.1.5 Define D-response token-give. This parameter specifies the code(s) which may be sent by the terminal to explicitly return the data token to the service following reception of a data token flag (see section 2.4). The redefinition has immediate effect. Default is '8'.

2.1.6 Define general receive inactivity timeout. This parameter specifies the value of the general receive inactivity timeout (see section 5.3). Default is 30s.

2.1.7 Define poll timeout. This parameter specifies the value of of poll timeout (see section 5.3). Default is 30s.

D-Set mode (mode other than mode 0) may be followed by a TDU or TDU's of up to 255 bytes in total length.

2.2 D-Control

D-Control allows selective and non-disruptive reset of DDU variables. D-Control carries a sequence code and may optionally carry any of the following parameters:

2.2.1 Mode. As specified in 2.1.1 above, but default is no change.

2.2.2 Define D-response positive. As specified in 2.1.3 above, but default is no change.

2.2.3 Define D-response negative. As specified in 2.1.4 above, but default is no change.

2.2.4 Define D-response token-give. As specified in 2.1.5 above, but default is no change.

2.2.5 Define general receive inactivity timeout. As specified in 2.1.6 above, but default is no change.

2.2.6 Define poll timeout. As specified in 2.1.7 above, but default is no change.

2.2.7 Reset. This parameter enables selective reset of the expected sequence number to 0, and of the code(s) for D-response positive and D-response negative to default. Default is no reset. In case of conflict with the redefinition parameters (sections 2.2.2 and 2.2.3 above), the most recently received parameter takes precedence. In the case of reset of D-response negative, the reset takes effect after a D-End group with a poll or data token flag set has been accepted (as in section 2.1.3 above).

If D-Control resets the expected sequence number it must either be the first processable-data VPDE on the videotex frame, or be immediately followed by the last D-End group on the videotex frame.

2.3 D-Data

This DDU carries a sequence code and is followed by a TDU or TDU's of up to 1023 bytes in total length.

2.4 D-End group

D-End group may directly follow one or more other processable-data VPDE's, or it may be transmitted alone (for example, to prevent an error occurring if more than 511 bytes of non-processable data are to be transmitted).

This VPDE carries the following control information.

- more flag, indicates, if set, that this is not the last processable-data VPDE in the videotex frame; further processable-data VPDE's should be expected without request

- poll flag, indicates, if set, that no further processable-data VPDE's will be sent until requested, and that the terminal is expected to transmit a D-response positive if this block and all previous blocks have been received correctly. (Note: the response to a poll flag does not necessarily imply that any associated TDU's have been acted on or accepted.)

- data token flag, indicates, if set, that no further processable-data VPDE's will be sent until requested, and that the terminal is expected to return the data token after this block and all previous blocks have been received correctly and processed by the application. The data token may be returned either implicitly by transmission of any data by the application, or explicitly by transmission of D-response token-give.

(Note: only one of the three above flags may be set in a given D-End group.)

- discard flag, indicates that subsequent data up to the next processable-data VPDE delimiter is to be discarded.

2.5 D-U-Abort.

This DDU carries a sequence code, and may optionally be followed by a TDU or TDU's of up to 255 bytes in total length. It may be transmitted by the service to indicate that the use of processable-data VPDE's is to be abruptly terminated.

3 FUNCTIONS OF TELESOFTWARE DATA UNITS

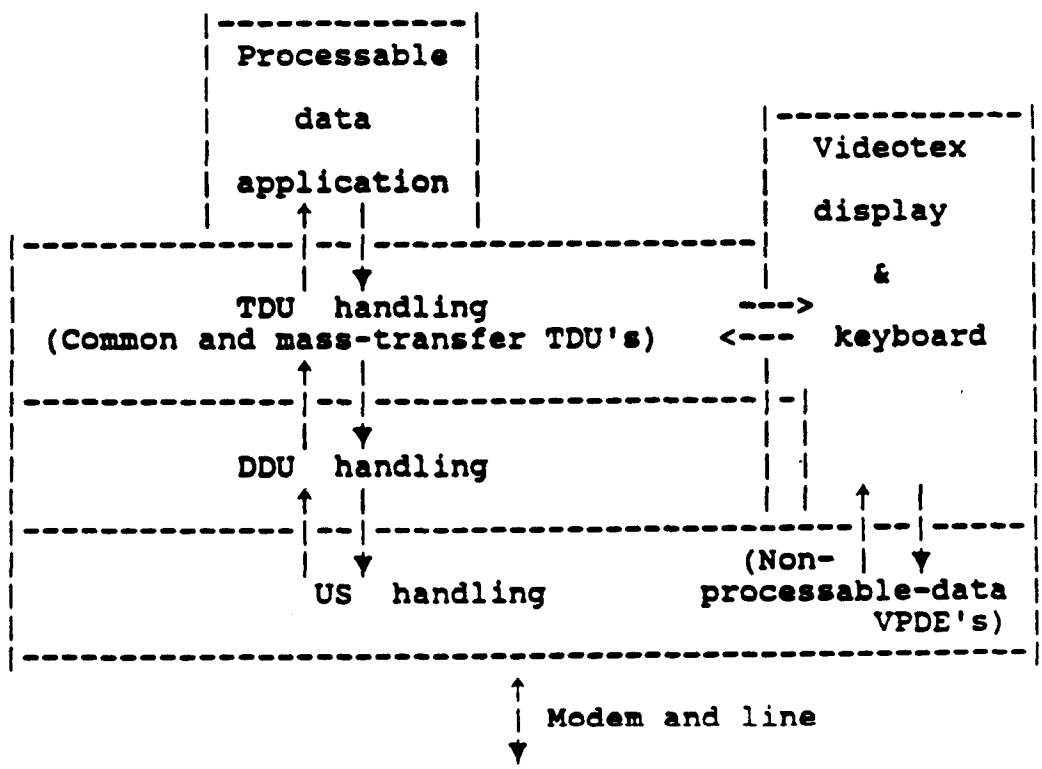
3.1 Streams

The use of TDU's enables communication between a single line connection and one or two independent 'streams'. They also enable control of the interaction between the application and the videotex display.

Figure 3 shows a model of the flows of data within a theoretical processable-data terminal. In this model, non-processable-data VPDE's are directed immediately to the display, while processable-data VPDE's are first examined by a 'DDU handler'; this handler may also perform 7-to-8 bit conversion before passing the TDU's to a 'TDU handler'. The TDU handler may be regarded as controlling the data flow between three entities, the line/DDU handler, the display/keyboard, and the processable-data applications. Keyboard data is shown as passing through the TDU handler since, while a processable-data application is associated, that application may wish to receive keyboard-entered data rather than it being transmitted directly to line.

Stream 0 is permanently associated with the standard videotex display and keyboard, while stream 1 may be associated with a processable-data application which may be available in a terminal. A single data unit may be addressed to more than one stream simultaneously.

FIGURE 3 - THEORETICAL MODEL OF PROCESSABLE DATA TERMINAL



3.1.1 Keyboard-entered data may either be forwarded immediately to the service, or be processed within the terminal, according to the requirements of the terminal software.

3.1.2 Non-processable-data VPDE's should be passed directly to the videotex display.

Stream 0 data (ie data within processable-data VPDE's intended for the display) should be interpreted as in a 8-bit environment, regardless of the environment of other display data. However, 'ESC 4/x' and 'ESC 5/x' should be accepted as valid codings of C1 controls, in addition to the 8-bit codings '8/x' and '9/x'.

3.2 Terminal parameters

The operation of the processable-data facility is controlled by the five following parameters. Each of these parameters defaults on reception of a D-Set mode DDU.

3.2.1 Display access inhibit flag.

This flag may be set to indicate that data generated by a user application and intended for the display may not be passed to the display until all stream 0 data up to the following data token flag has been passed to the display. If this flag is not set, it is the responsibility of the service and/or terminal software to ensure that the possible interleaving of blocks of data from the service and terminal software respectively does not give rise to undesirable effects.

Default is flag unset, which is additionally restored on reception of the data token flag.

3.2.2 Videotex command mode flag.

In this subset, this flag must be set before or at the time that any application is associated with stream 1, to indicate that the service expects to receive videotex commands from the terminal. (The default value (unset) is reserved for future extensions in which terminals may themselves transmit processable-data VPDE's to the service.)

3.2.3 T-Association reject

This is a variable-length string which is to be transmitted to reject a T-Associate, or to abort the processable data facility. (Default is the single-character string '9'.)

Note: this response is defined independently of the responses defined by DDU's.

3.2.4 Application Response Timeout

This timeout indicates the maximum time which should elapse after the terminal has sent an application response before further data is received from the service (see section 6.1.3). The default value is 30s.

4 REPERTOIRE OF TELESOFTWARE DATA UNITS

This section describes the TDU's and their parameters (see also Table 3).

TABLE 3 - TELESOFTWARE DATA UNITS

TDU	Parameters	Mandatory /optional
Common		
T-Control	Display access inhibit flag	O
	Videotex command mode flag	O
	New T-Association reject	O
	Application response timeout	O
T-Associate*	Stream	M
	Application name	M
	Association identifier	O
	Optional subset	O
	Display access inhibit flag	O
	Videotex command mode flag	O
	New T-Association reject	O
	Application response timeout	O
T-Release*	Stream	M
T-Dissociate*	Stream	M
T-U-Abort*		
T-Data**		
Mass Transfer		
T-Write-Start***	Stream	M
	Transfer identifier	M
	Data structure	O
	Relative address	O
T-Write**	Stream	M
	Transfer identifier	O
	Relative address	O
T-Write-End***	Stream	M
	Transfer identifier	M
	Relative address	O
T-Write-Restart	Stream	M
	Transfer identifier	M
	Data structure	O

/cont

TABLE 3 (cont)

Telesoftware-Specific

T-Capability-Spec	Stream	M
	Target machine	O
	Peripheral	O
T-Filespec***	Stream	M
	Destination-code	O
	Destination-name	O
	Filename	O
	Date/time of last modification	O
	New/amend/extend-indicator	O
	File-length	O
	File-type	O
	Text coding	O
	Encryption-related-data	O
	Load-address	O
	Execute-address (absolute)	O
	Execute-address (relative)	O
	Access-rights	O
	Usage-rights	O
	Transfer-identifier	O
T-Instruction**	Stream	M
	Language	O
T-Give-Control	Stream	M
	Download	O

Auxiliary-Device-Specific

T-Capability-Spec	Stream	M
	Device	M
T-Transfer-Spec***	Stream	M
	Transfer-length	O
	Encryption-related-data	O
	Transfer-identifier	O

* This TDU may be followed by a further TDU (subject to Table 6)
** This TDU header must be followed by data
*** This TDU header may be followed by data

4.1 Common TDU's

4.1.1 T-Control

T-Control is transmitted by a service to selectively set some of the terminal parameters (see section 3.2). These parameters are changed immediately the T-Control TDU is received. The display access inhibit flag and the videotex command mode flag may not be changed between reception of any VPDE which affects a stream other than stream 0 and reception of a data token flag.

4.1.2 T-Associate

T-Associate is used by a service to request a terminal to associate stream 1 with a specified processable-data application (identified by the 'Application name' parameter).

T-Associate has two mandatory parameters, 'stream' (set to 1), and 'application name' (see 4.3 and 4.4 below). The optional parameter, 'optional subset' may be used to identify the intended use of optional syntax elements; at present only the mass-transfer subset (see 4.2 below) has been defined.

A T-Associate TDU may also carry an 'association identifier' parameter, and/or be followed by a T-Capability-Spec or T-Filespec TDU. The 'association identifier' provides a unique label which unambiguously identifies the processable-data application in that videotex service, although it is not necessarily unique to an individual user or to a particular occasion. It enables that application to be identified if it is necessary to re-enter it (for example, to re-attempt or repeat the intended transaction). This parameter includes a prefix to indicate whether it may be used as, or as part of, a videotex command to re-enter the same application at a later time.

The specification of an application may include values of the terminal parameters (section 3.2), and of the optional subset parameter, to be implicitly invoked when the application is associated. In addition, any of the parameters permitted with a T-Control TDU may be included in a T-Associate TDU, subject to the provisions of 4.1.1; such explicit parameter values take precedence over any values implicitly invoked by the application.

4.1.3 T-Release

A T-Release TDU is sent by a service to request the orderly termination of the processable-data application. It carries a 'stream' parameter (set to 1) and may also be followed by a T-Instruction TDU.

4.1.4 T-Dissociate

A T-Dissociate TDU may be sent by a service to request forced termination of the processable-data application without ending the use of processable-data VPDE's. It carries a 'stream' parameter only. This TDU may, in the general case, be followed by a further TDU; this facility is not used by the applications described in this document.

4.1.5 T-U-Abort

A T-U-Abort TDU may be sent by a service to request forced termination of the processable-data application and at the same time ending the use of processable-data VPDE's. It carries no parameters. This TDU may, in the general case, be followed by a further TDU; this facility is not used by the applications described in this document.

4.1.6 T-Data

A T-Data TDU may be sent by a service to carry data intended for display alone.

4.2 Mass Transfer TDU's

This set of TDU's provides for confirmed and reliable transfer of data to virtual memory space in the terminal. Its intended use must be declared by the use of the 'optional subset' parameter with the T-Associate TDU.

The virtual memory is regarded as a simple sequence of bytes (eight-bit groups), or as a sequence of records. The relationship of this virtual memory to files, foreground memory or other devices in the actual terminal is defined in each case by the corresponding T-Filespec TDU.

4.2.1 T-Write-Start

A T-Write-Start TDU may be sent by a service to announce commencement of a mass transfer. It has one mandatory parameter, transfer identifier. It also has an optional parameter, data structure, which indicates whether the data will be byte-structured or record-structured. Default is byte-structured.

This TDU may also carry a relative address parameter and/or data. These have the same effect as if they were carried by a T-Write TDU (section 4.2.2).

4.2.2 T-Write

A T-Write TDU may be sent by a service to carry data forming apart of the mass transfer. A number of T-Write TDU's may follow a T-Write-Start. In addition to the data, a relative address and/or transfer identifier parameter may be present in the TDU.

The data associated with each of these TDU's is by default intended to immediately follow that of the previous T-Write or T-Write-Start TDU in the virtual memory space, unless a 'relative address' parameter is used to identify the start address of the current block. The default relative address for the first data in a transfer is the first byte or record. In the case of record-structured transfer, the coding of the user data is specified in section 8.1.4.

4.2.3 T-Write-End

A T-Write-End TDU is sent by the service to indicate the end of a mass transfer. This TDU must carry a transfer identifier parameter, and may also carry data and/or a relative address parameter. These have the same effect as if they were carried by a T-Write TDU (section 4.2.2).

4.2.4 T-Write-Restart

A T-Write-Restart TDU may be sent by a service to announce the restart of a mass transfer. The transfer identifier parameter is mandatory, and a data structure parameter must also be present if the data is record-structured.

4.3 Telesoftware-specific TDU's

The use of this set of TDU's is announced by the application name '!T' (2/1, 5/4) or '!Tn' (2/1, 5/4, 3/n) in the T-Associate TDU to identify telesoftware downloading; 'n', if present, identifies the minimum terminal telesoftware profile (to be defined) required to complete the intended procedure. The mass-transfer subset is always required with this application.

4.3.1 T-Capability-Spec. This TDU may be transmitted by a service to interrogate the terminal concerning the capability of its main equipment (or operating system) and/or any peripheral equipment.

This TDU has two optional parameters, 'target machine' and 'peripheral'; at least one parameter must be present. The 'target machine' parameter may be used to identify characteristics of the central processing equipment, while one or more 'peripheral' parameters may be used to identify auxiliary equipment.

The terminal's response to a T-Capability-Spec TDU is not affected by any T-Capability-Spec TDU's which may already have been received during an association. That is, the most recently received T-Capability-Spec TDU supersedes all data in any previous VT-Capability-Spec TDU's.

4.3.2 T-Filespec. This TDU may be transmitted by a service to provide information on the characteristics of a file. The file may then either accompary the VT-Filespec TDU as a single user-data field, or the file may be transferrred subsequently using the mass transfer procedure.

The following optional parameters may be used:

4.3.2.1 Destination-code. This parameter may take the values 'don't care' (default), 'foreground memory', 'random-access background memory required', or 'cassette tape required'. (Note: the latter option may be required for some software to overcome operating system or security constraints.)

4.3.2.2 Destination-name. This parameter indicates a drive, device and/or directory name for the storage of the file. It may not be used in combination with a destination code. The length of the parameter is not limited by this protocol.

4.3.2.3 Filename. This parameter provides a 'name' which may be associated with the transferred file (for example, in the destination filestore; this filename is not necessarily related to any transfer identifier). The length of this parameter is not limited by this protocol. (It is recommended that the first 6 characters of the filename should be alphabetic or numeric, and that they should define a unique filename within the application context when alphabetic case is disregarded.) Drive, device or directory designations should not be included in this parameter, but file-type suffices may be included at the discretion of the service.

4.3.2.4 Date/time of last modification. This parameter indicates data and time of last modification of the file. The significance of the date and time is not guaranteed by this protocol.

4.3.2.5 New/amend/extend-indicator. This parameter indicates whether the downloaded data is intended to form a previously non-existing ('new') file (default), or to modify an existing file. In the case of 'amend', the address space for the transfer starts at the beginning of the old file, but may extend beyond the end of the old file; therefore by default the new data will overwrite the old file. In the case of 'extend', the address space for the transfer starts at the end of the old file. (It is the responsibility of the terminal software to warn the user of possible corruption of existing files, when appropriate.)

4.3.2.6 File-length. This parameter indicates the expected size in bytes of the new or modified file after the downloading procedure, or of the extension in the case of the 'extend' option.

4.3.2.7 File-type. Possible values of this parameter include text, operating-system command sequence in text form, source code in text form, source code in tokenised form, intermediate code, object code, executable code, binary data, numeric data. This parameter may also indicate the language used for programme code. If this parameter has the value 'description file', then the file data is coded as specified in section 8.3.

4.3.2.8 Text coding. This parameter indicates the coding scheme used for file-types containing text.

4.3.2.9 Encryption-related-data. This parameter may be used to identify an encryption algorithm which has been applied to the file data, together with any related data. (For further study.)

4.3.2.10 Load-address. This parameter may be used to indicate an address in foreground memory at which the data should be loaded. In the case of transfer to foreground memory, this parameter provides a base address for the transferred data; in the case of transfer to background memory this parameter may be recorded as a file parameter according to the terminal's filing system. The relationship of this address to any terminal operating-system virtual memory addressing scheme is not specified by this protocol.

4.3.2.11 Execute-address (absolute). This parameter may be used to indicate an absolute address at which programme execution should start when the file is loaded into foreground memory. The relationship of this address to any terminal operating-system virtual memory addressing scheme is not specified by this protocol.

4.3.2.12 Execute-address (relative). This parameter may be used to indicate an address,relative to the beginning of the file after loading into foreground memory, at which programme execution should start.

4.3.2.13 Access-rights. This parameter may be used to indicate access rights to the downloaded file by the terminal user. Default is no restriction. Enforcement of these rights is not a responsibility of the protocol. (For further study.)

4.3.2.14 Usage-rights. This parameter may be used to indicate a maximum number of times which the downloaded file may be used, or a time period after which it may not be used. Default is no restriction. Enforcement of these rights is not a responsibility of the protocol. (For further study.)

4.3.2.15 Transfer-identifier. This parameter may be used to identify the specified file with a transfer executed under the mass-transfer protocol.

4.3.3 T-Instruction. This TDU may be transmitted by a service to request the terminal to execute the instruction sequence included in the user-data field. The commands may be coded in a virtual operating system language (specified in section 8.4), unless the 'language' parameter indicates that the terminal's native operating system language is used.

4.3.4 T-Give-Control. This TDU may be transmitted by a service to offer the terminal the task of initiating file transfers. The 'download' parameter indicates whether, if the terminal responds by returning the data token to the service (D-response token-give), the service will proceed to download the file declared in the last T-Filespec TDU, or the file(s) declared in the last description file.

4.4 Auxiliary-device-specific TDU's

The use of this set of TDU's is announced by the application name '!A' (2/1, 4/1) or '!An' (2/1, 4/1, 3/n) in the T-Associate TDU to identify data transfer to an auxiliary device; 'n', if present, identifies the minimum terminal auxiliary-device profile (to be defined) required to complete the intended procedure. The mass-transfer subset may be used with this application.

4.4.1 T-Capability-Spec. This TDU may be transmitted by a service to indicate, in either private or standardised format, information about the intended auxiliary device. The request TDU has one parameter, 'device'.

The terminal's response to a T-Capability-Spec TDU is not affected by any T-Capability-Spec TDU's which may already have been received during an association. That is, the most recently received T-Capability-Spec TDU supersedes all data in any previous T-Capability-Spec TDU's.

4.4.2 T-Transfer-Spec. This TDU may be transmitted by a service to provide a declaration of the parameters of an intended transfer.

The following optional parameters may be used:

4.4.2.1 Transfer-length. This parameter may be used to indicate the approximate expected length of the transfer.

4.4.2.2 Encryption-related-data. This parameter may be used to identify an encryption algorithm which has been applied to the data for transfer, together with any related data. (For further study.)

4.4.2.3 Transfer-identifier. This parameter may be used to verify the association between the T-Transfer-Spec and a subsequent transfer executed under the mass-transfer protocol.

If the mass-transfer option has not been invoked with the T-Associate TDU, then the data to be transferred to the auxiliary device immediately follows the parameter field of the T-Transfer-Spec TDU.

The data associated with each T-Transfer-Spec is intended to be regarded as a complete entity independent of other data transfers.

5 USE OF DIALOGUE DATA UNITS

The provisions of this section are subject to the provisions of Annex A, section A.1, if BCS are in use.

Where recovery procedures from error situations are identified, practical terminal implementations will need to limit the number of retries which may be attempted.

5.1 Invocation of Processable Data Facility

While in mode 0, the only valid DDU is a D-Set mode proposing a mode other than 0. Other DDU's (and any accompanying TDU's) should be ignored. If the terminal is able to accept the D-Set mode DDU it should change to the specified mode and continue to decode subsequent data accordingly.

If a terminal is able to interpret a D-Set mode but unable to accept a proposed parameter in it, then the terminal should transmit a D-response mode-reject and return to mode 0.

5.2 Operation while using the Processable Data Facility

Sequence codes should be checked and complete Processable Data VPDE's should be acted on in the order in which they are received.

Following correct receipt of a D-End group, action must be taken according to the flags set as described in 2.4 above. The terminal may delay transmission of a response if it is necessary to wait for internal buffers to become available.

If, during reception of a processable-data VPDE, a VPDE other than a processable-data VPDE is received (that is, a single 'US' (1/15) character followed by any character other than 3/14), then action should be taken as if a D-End group with no flags set had been received immediately preceding it.

5.2.1 Action in the event of DDU errors

In the event of the following DDU reception errors, the terminal should transmit a D-response negative to request re-transmission:

- a numbered DDU with a sequence code out of order
- an unnumbered DDU between a numbered DDU and a D-End group with a poll or data token flag set
- a sequence code (other than 4/0) more than once between the reception of successive D-End groups with either the 'more' or 'poll' flag set
- TDU(s) in excess of permitted length
- more than 511 bytes following a D-End group with any flag set before a further DDU delimiter (the two bytes of the delimiter being included in the count of 511 bytes)
- BCS error (see Annex A)

Detailed DDU error-recovery strategy is a matter of terminal design not determined by this Recommendation: a terminal may, for example:

- buffer all received VPDE's until a D-End group without the more flag set is received before processing them or responding to any errors; in the event of a DDU error this terminal would discard the buffered VPDE's, transmit a D-response negative and reset its expected sequence number to its previous value, or

- process received VPDE's as soon as they are complete; in the event of a DDU error this terminal may transmit a D-response negative immediately, and discard subsequent received data (including any received D-End group) until it received a VPDE with a sequence code which may validly follow the last processed VPDE.

(Note: The second strategy outlined above may occasionally cause undetected loss or repetition of an unnumbered DDU, or of a D-End group either on its own or following an unnumbered DDU. It is a responsibility of the service to ensure that this would not cause unacceptable application errors.)

5.2.2 Action in the event of DDU exceptions

The following exceptions should not cause transmission of a D-response negative, but may cause higher-level error recovery to be initiated (see sections 6.2.1 and 6.3.1 for examples):

- reception of a D-Set mode (except mode=0) DDU (unless received in the course of attempting higher-level error recovery)

- unacceptable parameters or parameter values in a received D-Control DDU

- protocol error (for example, an unrecognised DDU command identifier)

- excessive retransmissions of D-response negative

5.2.3 Action in the Event of Higher-Level Transmission

Transmission of data by the T-level (for example, the character strings specified in section 8.6, or the strings defined by the Application or Transfer Identifiers) implicitly returns the data token if it is held by the terminal. The response by the service to a transmission by the terminal at T-level must commence with an unnumbered DDU.

Therefore in the event of transmission of data by the T-level, the D-level should discard subsequent received data (including any received D-End group) until it receives an unnumbered VPDE.

5.3 Release of Processable Data Facility

While the processable data facility is invoked, its orderly release may be requested by the service by transmission of a D-Set mode or a D-Control specifying mode 0. The terminal may then transmit a D-response positive and enter mode 0.

If a D-U-Abort DDU is received, any associated TDU should be acted on and the terminal should immediately return to mode 0. No response is required from the terminal.

5.4 Timers

If one of the following timers expires, the terminal may attempt recovery by issuing a D-response negative. If this fails to achieve recovery after a limited number of attempts, then the error recovery procedures of section 6.3.1 should be initiated.

5.4.1 General receive inactivity timer. This timer is only active while the mode is not 0. It is started by the reception of a processable-data delimiter and restarted by the reception of any data. It is stopped by the reception of a D-End group, unless that D-End group has the 'more' flag set, or by the transmission of a D-response negative. (It is for further study whether this timer should take more precise account of any period when the terminal is causing a lower layer (such as a link layer) to exercise flow control.)

5.4.2 Poll timer. This timer is started by the transmission by the terminal of a D-response positive or of a D-response negative. It is stopped by the reception of a DDU with a sequence code indicating either the required sequence number or an unnumbered block. (It is for further study whether this timer should take more precise account of any period when the terminal is causing a lower layer (such as a link layer) to exercise flow control.)

6 USE OF TELESOFTWARE DATA UNITS

6.1 General Principles of Operation

6.1.1 Rules of Operation of the Processable-data Facility

A T-Control TDU may be received at any time, whether or not an application is associated.

If a TDU immediately following a DDU is addressed to stream 0, then all data in that VPDE following that parameter field should be passed to the videotex display. (Note: this data may be interleaved with non-processable-data VPDE's which have been directed immediately to the display (section 3.1).)

No TDU or data may be passed to stream 1 until a T-Associate has been received. If the application specified in a T-Associate TDU is not available in the terminal then the terminal should transmit a T-Association reject.

The service may request termination of the application by means of the T-Release TDU. The terminal may then implicitly accept the release, providing there is a preceding D-Set mode DDU or a following D-End group DDU, by transmitting a positive response to that DDU.

Reception of a T-Dissociate or T-U-Abort, or transmission of a T-Association reject, causes the local application to be terminated immediately; any subsequent TDU's directed to stream 1 should be discarded until a further T-Associate TDU is received. (A terminal may indicate at any time that it wishes to discontinue the association by transmitting a T-Association reject. Note: transmission by the terminal of a T-Association reject is not sufficient to terminate the use of processable-data VPDE's.)

6.1.2 Use of the Data Token Flag

The purpose of the data token flag is to enable the service to ensure that it has received a response to all TDU's transmitted up to that time, before the data token is returned to the service.

(Note: If the terminal is unable to accept a TDU, or if an error or DDU exception occurs, the character strings specified in section 8.6, or a string defined by an association or transfer identifier, may be immediately transmitted to the service. Transmission of these strings implicitly returns the data token to the service (see section 5.2.3).)

6.1.3 Application Response Timer

The application response timer is started by the return of the data token, or by transmission of data other than a D-response. It is stopped by the reception of a TDU, or of a D-Set mode or D-U-Abort DDU. If the timer expires, then recovery may be attempted as specified in section 6.2.1 or 6.3.1 below.

6.2 Operation of the Mass Transfer Procedure

It is the purpose and responsibility of the mass transfer procedure to ensure the correct transfer of identified mass data. Mass transfer is only permitted during a 'mass transfer phase'.

Optionally, the terminal may request a mass transfer by transmitting the string defined by a transfer identifier.

A mass transfer phase is initiated by the reception of a T-Write-Start TDU. This causes a transfer identifier to be established for the current mass transfer phase, equal to that received with the T-Write-Start TDU.

During a mass-transfer phase, reception of a valid T-Write TDU (that is, one with no transfer identifier, or with a transfer identifier identical to that in the T-Write-Start) should cause the data to be added or overwritten at the appropriate place in the defined memory space.

The data token flag should always be received immediately following a T-Write-End TDU; a mass transfer phase is then normally terminated by transmission of a D-response token-give with no preceding data.

If a T-Write-Restart TDU is received with the same parameter values as those in the initial T-Write-Start TDU, then the mass transfer phase should be reset and continued.

6.2.1 Abnormal termination of mass-transfer

The following error conditions during a mass transfer phase cause immediate termination of the mass transfer:

- occurrence of a DDU exception condition (see section 5.2.2)
- reception of any TDU directed to stream 1 other than a T-Write or T-Write-End TDU
- reception of a T-Write TDU or T-Write-End TDU with a transfer identifier present, but which does not match that established when the phase was initiated
- expiry of the application response timer

In the case of any of these conditions, or in the case of a local error, then the terminal may either:

- if a transfer identifier of non-zero length (and with a prefix other than 2/0) has been established, transmit the string defined by the transfer identifier, else
- transmit a T-Read-Restart (section 8.6.2)

If this action causes the reception of a T-Write-Restart or T-Write-Start TDU with the same parameter values as those in the initial T-Write-Start TDU, then the mass transfer phase may be reset and continued from the beginning.

A mass transfer phase may also be abnormally terminated by:

- reception of a dissociate or abort DDU or TDU

- transmission by the terminal of any sequence other than a D-response (for example, the terminal may indicate at any time that it wishes to discontinue the application association by transmitting a T-Association reject, or that it wishes to discontinue the transfer by transmitting a T-Application-Reject)

6.2.2 Errors outside a mass transfer phase

The following error conditions may occur outside a mass transfer phase:

- reception of a T-Write TDU

- reception of a T-Write-End TDU

- reception of a T-Write-Restart TDU

In these cases the incorrect received TDU should be discarded. Recovery from repeated occurrence of these error conditions may be attempted using the application-level procedures (see, for example, section 6.3.1.).

6.3 Operation of Telesoftware Downloading

Telesoftware-specific TDU's may be transmitted at any time outside a mass transfer phase.

A terminal may reject a T-Capability Spec or a T-Filespec TDU, or indicate failure to implement a T-Instruction TDU, at any time by transmitting a T-Application-Reject.

If while a T-Filespec TDU has been received but the corresponding file has not been received, a second T-Filespec TDU is received, then the information in the first T-Filespec TDU should be discarded. A Filespec segment in a description file may not have the same transfer identifier as that of a previous T-Filespec TDU. If a T-Filespec TDU is received with the same transfer identifier as a Filespec segment in a description file, then the parameters of both the TDU and the segment apply to the file; if any parameters conflict, then the values in the TDU take precedence.

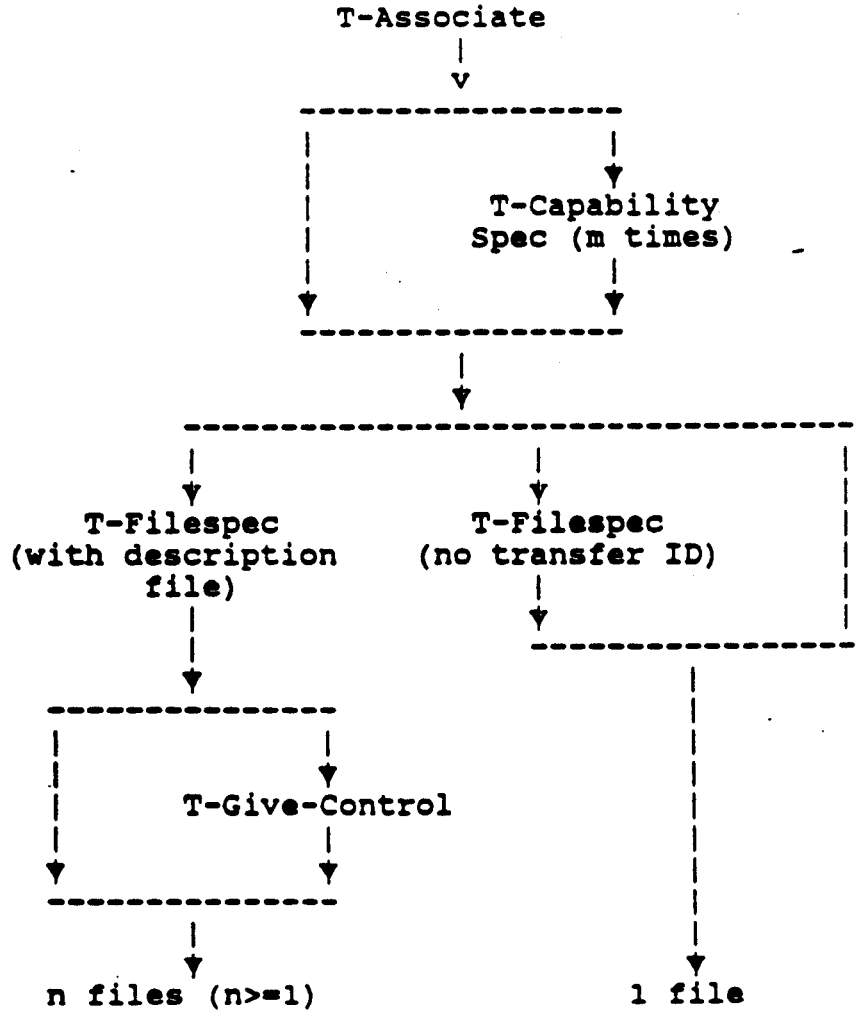
During a mass transfer phase, data fields accompanying T-Write or T-Write-End TDU's are assembled into a telesoftware file according to the parameters specified in the corresponding description file Filespec segment and/or T-Filespec TDU. The data token flag should always be received immediately following a T-Write-End TDU or a T-Filespec TDU with a user-data field); a D-response token-give should only be transmitted when the terminal has successfully received and stored (as required) the transferred file.

The data token flag should always be received immediately following a T-Give-Control TDU; the terminal may then respond by simply transmitting a D-response token-give (if the 'download' parameter is set to its default value), or may transmit the sequences defined by the transfer identifier parameters, to initiate the required file transfers.

If a T-Instruction TDU is received immediately following a T-Release TDU, then the release should only be accepted after the instructions have been implemented.

FIGURE 4 - POSSIBLE SEQUENCES OF TDU'S PRECEDING A TELESOFTWARE FILE DOWNLOAD

(Note: this figure does not show all the possible sequences of TDU's which may be used to download a set of files.)



6.3.1 Error Recovery

If an error condition occurs during telesoftware downloading, other than a mass transfer error from which recovery is achieved using the procedures of 6.2 above, and if an association identifier of non-zero length (and with a prefix other than 2/0) has been established, the terminal may attempt recovery (up to a limited number of retries) to the start of the association. If this procedure causes the reception of a T-Associate TDU with the same association identifier, then the downloading may be restarted.

6.4 Operation of Auxiliary-Device Transfer

Auxiliary-device-specific TDU's may be transmitted at any time outside a mass transfer phase. Figure 5 illustrates possible sequences of TDU's to effect an auxiliary-device transfer.

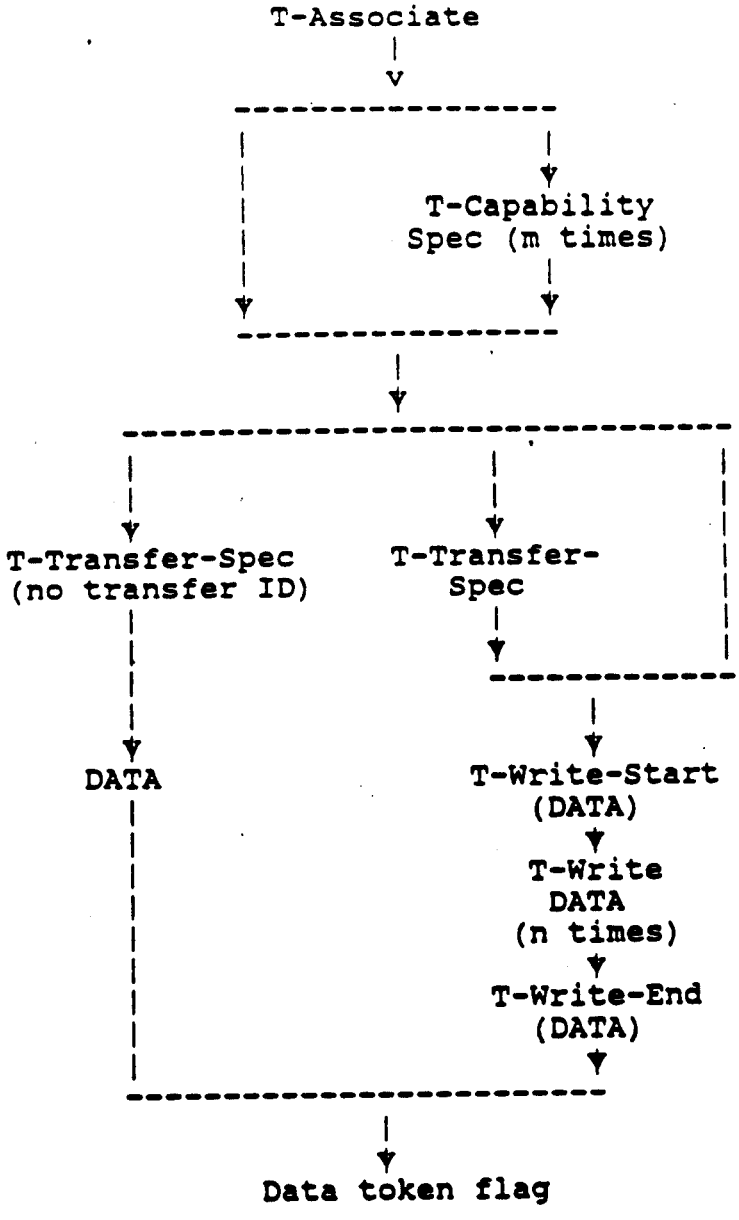
A terminal may reject a T-Capability-Spec or a T-Transfer-Spec TDU at any time by transmitting a T-Application-Reject.

During a mass transfer phase, data fields accompanying T-Write or T-Write-End TDU's are passed to the auxiliary device. The data must be transferred sequentially and contiguously; the 'relative address' parameter may only be used to verify the sequence of the data, not to alter it.

The data token flag should always be received immediately following a T-Transfer-Spec TDU followed by a data field. A D-response token-give should only be issued in response to this TDU, or to a T-Write-End TDU, when the terminal has successfully received and forwarded the transferred data, or has reliably stored it for later transfer.

6.4.1 Error recovery; as for telesoftware downloading (see section 6.3.1).

FIGURE 5
POSSIBLE SEQUENCES OF TDU'S TO EFFECT AUXILIARY-DEVICE TRANSFER



7 CODING OF DIALOGUE DATA UNITS

The coding of DDU's depends on the class of the DDU. The coding structures of the three classes are illustrated in Figure 6, and examples are shown in Figure 7.

In order to provide compatibility with both 7- and 8-bit systems, all delimiters, and command and parameter identifiers, are specified as 7-bit codes in the range 0/0 to 7/15 The value of the most-significant bit should be ignored in interpreting these codes.

FIGURE 6 - DDU CODING STRUCTURE

a) DDU Commands

Field	PD VPDE Delimiter	Command identifier	Sequence code	Parameter field length indicator	Parameter field
Mnemonic		CI	SC	LI	
No. bytes	2	1	1	1	n
Codes	1/15 3/14	2/x	4/x or 5/x	4/x, 5/x, 6/x or 7/x	see below

TDU1 . .

The parameter field comprises 0, 1 or more parameters, each coded:

Parameter identifier	Parameter value length indicator	Parameter value
PI	LI	PV
1	1	n
2/x	4/x, 5/x, 6/x or 7/x	any code string

b) D-Data

PD VPDE delimiter	Sequence code
	SC
2	1
1/15 3/14	4/x or 5/x

TDU1 . .

c) D-End group

PD VPDE delimiter	Flags	Checksum
		BCS
2	1	3
1/15 3/14	3/x	4/x, 5/x 6/x or 7/x

7.1 Processable-data Delimiter

DDU's always commence with the processable-data delimiter: the sequence 1/15, 3/14. The type of DDU (D-Command, D-Data or D-End Group) is determined by the character immediately following the delimiter:

7.2 DDU Commands

These are identified by a command identifier (CI) from column 2 of the code table immediately following the delimiter. The command identifiers are listed in Table 4.

The CI is followed by a sequence code taken from columns 4 or 5 of the code table. The sequence number is calculated modulo 31 (1Fhex) and is coded by adding the value 65 (41hex). Sequence codes therefore run 4/1, 4/2, . . .4/15, 5/0, 5/1, . . .5/15, 4/1, 4/2, . .etc. The code 4/0 indicates an unnumbered block; in this case there is no sequence number and the expected sequence number is not incremented. This code must be used with a D-Set mode (mode other than 0), but may also be used with any other block where sequence checking is not required.

The sequence code is then followed by a length indicator (LI) taken from columns 4-7 of the code table. The least significant six bits of the LI indicate the length of the following parameter field in bytes as transmitted. The use of the code 7/15 as a length indicator is reserved for possible future extension.

The parameter field comprises groups of:

- Parameter Identifier (PI). A single byte identifying the parameter. The coding of PI's is specified in Table 5.

- Length Indicator (LI). This length indicator is used to indicate the length of the following parameter value in bytes as transmitted. This LI is coded as specified above for the parameter field LI.

- Parameter Value (PV). The coding of parameter values is specified in 7.2.1 below. Note that all variable-length command parameters are subject to any 7-8 bit translation in effect, including any translation which may have been specified earlier within the same parameter field.

No parameter identifier may occur more than once in any one DDU.

7.2.1 Coding of DDU command parameters

7.2.1.1 Define Mode. This parameter is coded in a single byte:

4/0 no use of processable-data VPDE's
4/1 no translation performed (see 1.1)
4/2 '3-in-4' coding (see 1.2)
4/3 the 8-bit shift scheme described in 1.3
4/4 the 7-bit shift scheme described in 1.4

7.2.1.2 Reset. This parameter is a single byte in the range 4/0 to 4/7 where the following bits are significant:

bit 0 = 1 reset sequence number
bit 1 = 1 reset Dresponse positive
bit 2 = 1 reset Dresponse negative

Default is 4/0 (no reset)

7.2.1.3 Define D-response positive. This parameter is a variable length string of up to 16 bytes which is the new coding to be used by the terminal for a D-response positive.

7.2.1.4 Define D-response negative. This parameter is a variable length string of up to 16 bytes which is the new coding to be used by the terminal for a D-response negative.

7.2.1.5 Define D-response mode-reject. This parameter is a variable length string of up to 16 bytes which is the new coding to be used by the terminal for a D-response mode-reject.

7.2.1.6 Define D-response token-give. This parameter is a variable length string of up to 16 bytes which is the new coding to be used by the terminal for a D-response token-give.

TABLE 4
CODING OF DDU COMMAND IDENTIFIERS

DDU	CI
D-Set mode	2/7
D-Control	2/5
D-Data	see text (section 7.3)
D-End group	see text (section 7.4)
D-U-Abort	2/9

Note: CI's with bit 0 = 0 are reserved.

TABLE 5
CODING OF DDU PARAMETER IDENTIFIERS

Parameter	PI
Sequence code	see text
Define checksum use & Mode	2/2
Reset	2/6
Define D-response positive	2/1
Define D-response negative	2/5
Define D-response mode-reject	2/7
Define D-response token-give	2/13
Define general receive inactivity timeout	2/8
Define poll timeout	2/12
Flags	see text
BCS	see text

7.2.1.7 Define general receive inactivity timeout. This parameter is a single byte in the range from 4/1 to 7/15. The timeout interval is coded in binary form in the six least significant bits, in increments of 1s. (The possible use of 4/0 is for further study.)

7.2.1.8 Define poll timeout. This parameter is a single byte in the range from 4/1 to 7/15. The timeout interval is coded in binary form in the six least significant bits, in increments of 1s. (The possible use of 4/0 is for further study.)

7.2.2 TDU

A TDU or TDU's of up to 255 bytes may follow the parameter field.

7.3 D-Data

This DDU is identified by a sequence code from column 4 or 5 of the code table immediately following the delimiter (see section 7.2). The sequence code is immediately followed by a TDU or TDU's of up to 1023 bytes.

7.4 D-End group

This block is identified by the D-Flags parameter taken from column 3 of the code table immediately following the delimiter.

The two least significant bits of D-Flags have the following significance:

b1	b0	
0	0	No flag set
0	1	More flag set
1	0	Poll flag set
1	1	Data Token flag set

The third least significant bit (b2) is set if the discard flag is set.

7.5 Coding of Responses by the Terminal

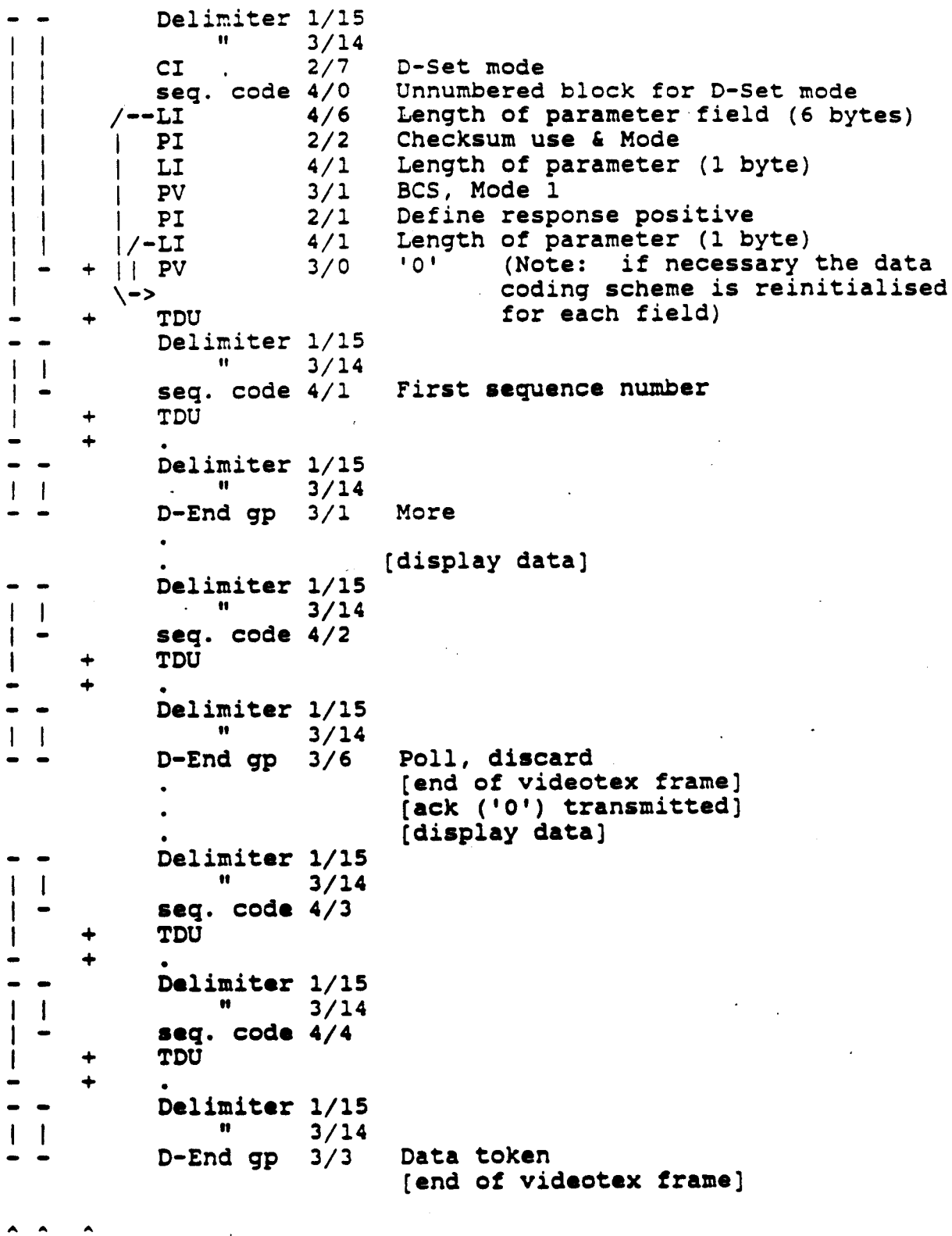
7.5.1 D-response positive. This response is coded as the single character 3/0 ('0') unless it has been redefined as specified in 7.2.1.3.

7.5.2 D-response negative. This response is coded as the single character 3/1 ('1') unless it has been redefined as specified in 7.2.1.4.

7.5.3 D-response mode-reject. This response is coded as the single character 3/9 ('9') unless it has been redefined as specified in 7.2.1.5.

7.5.4 D-response token-give. This response is coded as the single character 3/8 ('8') unless it has been redefined as specified in 7.2.1.6.

FIGURE 7
EXAMPLES OF DDU CODING



8 CODING OF TDU's

8.1 Structure of TDU's

Each TDU follows directly after a DDU or a preceding TDU and is coded according to the structure shown in Figure 8. Examples of TDU coding are shown in Figure 9. (Note that the whole TDU is subject to the 7-8 bit translation specified by the most recent DDU mode parameter.)

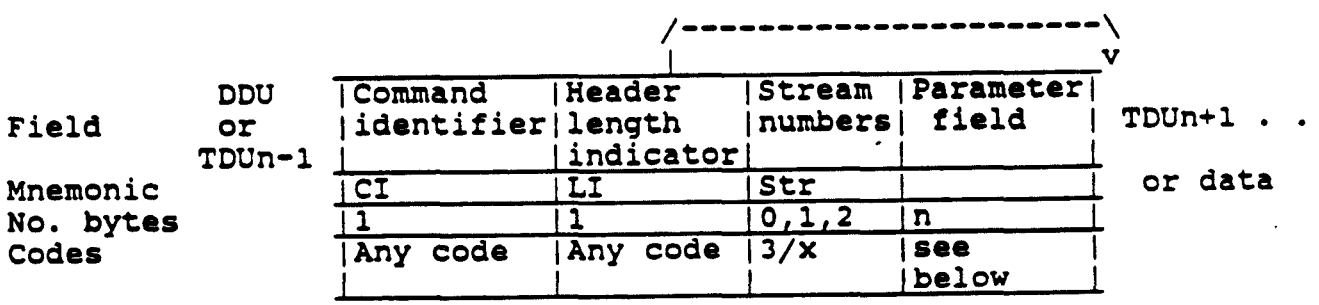
8.1.1 Command identifier (CI)

The first byte identifies the TDU according to the coding in Table 6. Columns 2 and 3 of the code table are reserved for common TDU's, and columns 4 and 5 are reserved for mass-transfer TDU's. Applications may use any other codes; the standardised applications in this document use column 6.

8.1.2 Length indicator (LI)

The length indicator is used to indicate the length of the parameter field of the TDU. The LI is coded as a single byte, and is the binary representation of the length of the subsequent field in bytes. The use of the code 15/15 as a length indicator is reserved for possible future extension.

FIGURE 8 TDU CODING STRUCTURE



The parameter field comprises 0, 1 or more parameters, each coded:

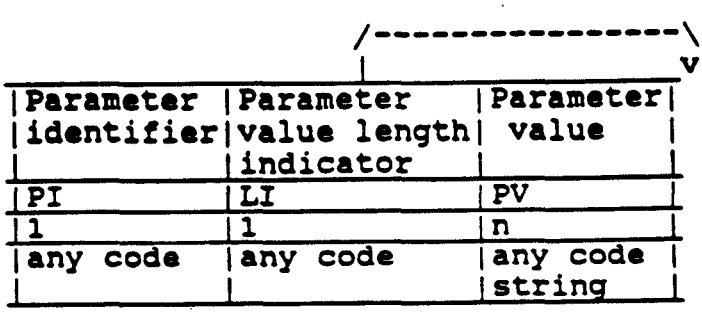


FIGURE 9 - Examples of TDU Coding

a)

```
CI  2/1   T-Control
LI  0/3
PI  4/0   Terminal flags
LI  0/1
PV  4/2   Command mode
```

b) (Two concatenated TDU's)

```
CI  2/3   T-Associate
LI  0/14
Str 3/1
PI  4/5   Application-name
LI  0/2
PV  2/1   ) Telesoftware
PV  5/4   )
PI  4/7   Association identifier
LI  0/4
PV  2/2   Page number
PV  3/1   1
PV  3/2   2
PV  3/3   3
PI  4/4   Optional subset
LI  0/1
PV  4/1   mass-transfer
CI  6/1   T-Capability-Spec
LI  0/11
Str 3/1
PI  6/1   Target machine
LI  0/8
PV  4/9   I
PV  4/2   B
PV  4/13  M
PV  5/0   P
PV  4/3   C
PV  4/0   @
PV  4/1   A
PV  5/4   T
```

c)

```
CI  4/5   T-Write
LI  0/2
Str 3/0
Str 3/1
Data
 .
 .
 .
```

8.1.3 Parameter field

The parameter field comprises zero, one or two stream numbers coded 3/0 or 3/1 (streams 0 or 1 respectively), followed by zero, one or more groups of:

- Parameter Identifier (PI). A single byte identifying the parameter. The coding of PI's is specified in Table 7. Parameters used by the common and mass-transfer TDU's are coded in column 4; the standardised applications in this document use columns 6 and 7.

- Length Indicator (LI). This length indicator is used to indicate the length of the following parameter value. This LI is coded as specified above for the parameter field LI.

- Parameter Value (PV). The coding of parameter values is specified in 8.2 below.

If no stream number is present the default is stream 0 alone.

8.1.4 Data

Data (if any) follows the parameter field. In the case of record-structured data, the record separator is coded 0/13, optionally followed by 0/10. These characters may not appear as data within the records. Records may span TDU boundaries, but this will cause an error if the new TDU has a relative address parameter (see section 8.2.8).

TABLE 6 - CODING OF TDU COMMAND IDENTIFIERS

TDU	CI	TDU may follow:
T-Control	2/1	D-Data or D-Set mode (mode ≠ 0)
T-Associate	2/3	D-Data or D-Set mode (mode ≠ 0)
T-Release	2/5	D-Data or D-Set mode (mode = 0)
T-Dissociate	2/9	D-Data or D-Set mode (mode = 0)
T-U-Abort	2/11	D-U-Abort
T-Data	2/7	D-Data or D-Set mode (mode ≠ 0)
T-Write-Start	4/3	D-Data
T-Write	4/5	D-Data
T-Write-End	4/7	D-Data
T-Write-Restart	4/13	D-Data
T-Capability-Spec	6/1	D-Data or T-Associate
T-Filespec	6/3	D-Data or T-Associate
T-Transfer-Spec	6/3	D-Data or T-Associate
T-Instruction	6/7	D-Data or T-Release
T-Give-Control	6/5	D-Data

Note: CI's with bit 0 = 0 are reserved.

TABLE 7
CODING OF TDU PARAMETER IDENTIFIERS

Parameter	PI
Terminal flags	4/0
New T-Association reject	4/3
Application response timeout	4/6
Stream	see text (section 8.1.3)
Application name	4/5
Association identifier	4/7
Optional subset	4/4
Relative address	4/13
Data structure	4/14
Transfer identifier	4/15
Target machine	6/1
Device	6/1
Peripheral	6/3
Status	6/0
Destination-code	6/2
Destination-name	7/9
Filename	6/5
Date/time of last modification	7/15
New/amend/extend-indicator	6/4
File-length	6/7
Transfer-length	6/7
File-type	6/9
Text coding	7/13
Encryption-related-data	6/11
Load-address	6/13
Execute-address (absolute)	6/15
Execute-address (relative)	7/11
Access-rights	7/1
Usage-rights	7/3
Download	6/6
Language	7/7

The coding of description files which may be transferred during telesoftware dowloading is specified in section 8.3.

The coding of virtual operating system commands which may be used with the T-Instruction TDU in telesoftware downloading is specified in section 8.4.

The coding of data intended for the standardised printer in auxiliary-device transfer is specified in section 8.5.

8.2 Coding of TDU parameters

8.2.1 Terminal flags. This parameter is a single byte 4/n, where n (0 - 3) is such that:

bit 0 = 1 if display access inhibit = set
bit 1 = 1 if videotex command mode = set

Default is no change.

8.2.2 New T-Association reject. This parameter is a variable-length string of up to 16 bytes comprising the bytes to be sent by the terminal to form a T-Association reject. Default is no change.

8.2.3 Application response timeout. This parameter is a single byte in the range from 4/1 to 7/15. The timeout interval is coded in binary form in the six least significant bits, in increments of 1s. The code 4/0 has the special significance of 'timer disabled'.

8.2.4 Application name. This parameter is a variable-length string containing an application name. The use of the initial character 2/1 ('!') is reserved for standardised applications. The standardised application name for telesoftware downloading is 2/1, 5/4 ('!T'), and for auxiliary-device transfer is 2/1, 4/5 ('!A').

8.2.5 Association identifier. This parameter is a variable-length string which comprises two parts, an 'association identifier prefix', and the association identifier itself.

The prefix is a single byte 2/n which is the first byte of the parameter field. n may take one of the following values:

0 identifier cannot be used in videotex command mode to re-obtain T-Associate TDU
1 identifier to be transmitted alone to re-obtain T-Associate TDU
2 identifier to be included in page-number request to re-obtain T-Associate TDU
3 identifier to be included in keyword request to re-obtain T-Associate TDU

The association identifier itself is a variable-length string of up to 16 bytes.

Note: the precise format of the string to be transmitted by the terminal in the case of prefixes 2/2 and 2/3 will depend on the characteristics of the service to which the terminal is connected.

8.2.6 Optional subset. This parameter is a single byte:
4/0 no optional subset (default)
4/1 mass-transfer subset

8.2.7 Relative address. This is a variable-length parameter identifying, if present, an offset from the start of the virtual memory space to which the accompanying user data should be transferred. The offset is coded in absolute binary form using all eight bits of each byte. If the parameter is coded in more than one byte, then the first byte to be transmitted contains the most significant bits.

If the data is byte-structured, then the offset is counted in bytes (octets); the first byte in the virtual memory space has an offset of zero.

If the data is record-structured, then the offset is counted in records and the specified record commences at the start of the data field of the TDU; the first record in the virtual memory space has an offset of zero.

8.2.8 Data structure. This parameter is a single byte:

4/0 byte-structure (default)
4/1 record-structure

8.2.9 Transfer identifier. This parameter is a variable-length string which comprises two parts, a 'transfer identifier prefix', and the transfer identifier itself.

The prefix is a single byte 2/n which is the first byte of the parameter field. n may take one of the following values:

0 identifier cannot be used in videotex command mode to initiate transfer
1 identifier to be transmitted alone to initiate transfer
2 identifier to be included in page-number request to initiate transfer
3 identifier to be included in keyword request to initiate transfer

The transfer identifier itself is a variable-length string of up to 16 bytes. A zero-length string is permitted.

8.2.10 Target machine. This parameter is a variable-length string, comprising one or more fields. Fields are separated by the character '@' (4/0). The first field should contain a proprietary name which is sufficient to distinguish the principal operating system or target machine; the use of other fields is unstandardised. The use of the initial character '!' (2/1) in each field is reserved for possible later standardisation.

8.2.11 Peripheral. This parameter is a variable-length string, comprising one or more fields. Fields are separated by the character '@' (4/0). The first field should contain a name which is sufficient to distinguish the peripheral type; the use of other fields is unstandardised. The use of the initial character '!' (2/1) in each field is reserved for possible later standardisation.

8.2.12 Destination-code. This parameter is a single byte:

4/0 don't care (default)
4/1 foreground memory
4/2 background memory - random access required
4/3 cassette tape required

8.2.13 Destination-name. This parameter is a variable-length string containing the destination name.

8.2.14 Filename. This parameter is a variable-length string containing the filename. None of the characters from 2/0 to 2/15 or from 3/10 to 3/15 in the code table may be included in the filename, except that the character '.' (2/14) may be used to delimit a filename suffix.

8.2.15 Date/time of last modification. This parameter is a variable-length string of an even number of digits, coded in the range 3/0 to 3/9, specifying the date and time in the format yymmddhhmmss. The string may be truncated in two's from the right to give the date (and time) to a lower degree of accuracy.

8.2.16 New/amend/extend-indicator. This parameter is a single byte:

4/0 new (default)
4/1 amend
4/2 extend

8.2.17 File-length and transfer length. This is a variable-length parameter in which the expected length in bytes is coded in absolute binary form using all eight bits of each byte. If the parameter is coded in more than one byte, then the first byte to be transmitted contains the most significant bits.

8.2.18 File-type. This is a variable-length parameter of which the first byte identifies the type of data as follows:

3/0 description file
4/0 text (default)
4/1 os command sequence in text form
4/2 source code in text.form
4/3 source code in tokenised form
4/4 intermediate code
4/5 object code
4/6 executable code
4/7 binary data
4/8 numeric data

Subsequent bytes, if present, identify by means of a text string a language (such as 'BASIC', 'P-CODE') or a target processor (such as '6502'). This may optionally be followed by a '/' (2/15) and an identification of the language dialect (such as '/MSDOS').

8.2.19 Text coding. For further study

8.2.20 Encryption-related-data. For further study

8.2.21 Load-address. This is a variable-length parameter in which the load address is coded in absolute binary form using all eight bits of each byte. If the parameter is coded in more than one byte, then the first byte to be transmitted contains the most significant bits.

8.2.22 Execute-address (absolute). This is a variable-length parameter in which the absolute execute address is coded in absolute binary form using all eight bits of each byte. If the parameter is coded in more than one byte, then the first byte to be transmitted contains the most significant bits.

8.2.23 Execute-address (relative). This is a variable-length parameter in which the relative execute address is coded in absolute binary form using all eight bits of each byte. If the parameter is coded in more than one byte, then the first byte to be transmitted contains the most significant bits.

8.2.24 Access-rights. For further study.

8.2.25 Usage-rights. For further study.

8.2.26 Download. This parameter is a single byte:

4/0 declared files will be downloaded by the service if a D-response token-give is returned (default)

4/1 the terminal must use the transfer identifier(s) to download the declared files

8.2.27 Language. This parameter is a single byte:

4/0 virtual operating system language (default)

4/1 terminal's native operating system language

8.2.28 Device. This parameter is a variable-length string, comprising one or more fields. Fields are separated by the character '@' (4/0). The first field should contain a name which is sufficient to distinguish the device type; the use of other fields is unstandardised. The use of the initial character '!' (2/1) in each field is reserved for standardised devices.

8.2.28.1 If the first field commences with '!P' (2/1, 5/0), then the required device is a printer, and the transfer will be record-structured. The coding of the data field is specified in section 8.5.

A third byte may be present in the first field to indicate the minimum printer character repertoire required, as follows:

4/1 codes 2/0 - 2/2, 2/5 - 5/10 and 6/1 - 7/10 of the videotex primary graphic set, and the control codes specified in section 8.5.1.

4/2 the codes specified for 4/1, plus the mosaic characters in the first supplementary mosaic set

4/3 the alphanumeric repertoire specified in section 2.1.1 of T/CD 6-1 Part 1

4/4 the alphanumeric and mosaic repertoires specified in sections 2.1.1 and 2.1.2 of T/CD 6-1 Part 1

4/5 the codes specified for 4/4, plus DRCS capability

4/15 private use

There may follow a string of arbitrary length comprising a sequence of codes of additional control and/or graphic characters or character sequences which are required to be reproduced (for example, national currency signs, accented characters, selected line-drawing characters, graphic renditions).

If the repertoire specification byte is 4/15, then subsequent characters may be used to define the repertoire, and possibly also the page format or special stationery to be used.

The second field, if present and of non-zero length, specifies the page format required. It may be either:

- a decimal number coded using the bytes 3/0 to 3/9, indicating that the printer must be able to print rows of at least this number of characters.

- a letter, indicating that each record is to be formatted to a locally-specified line length, line breaks being inserted in place of spaces, or following hyphens ('-', 2/13). Permitted letters are 'L' (4/12) indicating that text is to be left justified, and 'J' (4/10) indicating that text preceding an inserted line-break should be justified to span the available line-length.

Examples of device parameter values are given in Fig 10.

8.3 Description files

A description file may be used to pass information concerning the set of files required for a specified transaction and the processing capabilities required to make use of that set of files.

A single description file contains information relating to a single 'set' of files; this information is contained in one or more data 'segments', each segment being coded in the form of a TDU according to the coding scheme specified in sections 8.1 and 8.2. A description file may contain a Capability-Spec segment, followed by one or more Filespec segments.

If there is more than one Filespec segment in a description file, then each segment must have a different transfer identifier.

FIGURE 10
EXAMPLES OF DEVICE PARAMETER VALUES FOR A STANDARDISED PRINTER

2/1 5/0 !P	Printer, no characteristics specified
2/1 5/0 4/1 4/0 3/8 3/0 !PA@80	Printer with basic alphanumeric repertoire, minimum 80 columns width
2/1 5/0 4/1 10/4 4/0 4/12 !PA$@L	Printer with basic alphanumeric repertoire plus $, records to be locally formatted to paragraphs and left justified
2/1 5/0 4/1 0/8 9/11 3/4 6/13 !PD[BS][CSI]4m	Printer with full alphanumeric and mosaic repertoires, overstrike and underlining capabilities

The description of the task to the user is not explicitly included in the format of a description file (since interactive videotex procedures are available to do this), although the set of files might include a text file containing descriptive information..

(Note: the contents of the file do not form part of the telesoftware protocol, and no response should be returned by the terminal to the data segments contained in a description file.)

8.4 Virtual operating system language

This section specifies the coding which may be used for data in the T-Instruction TDU.

The user-data field associated with a T-Instruction TDU must be record-structured, following the rules specified for record-structured mass transfer (section 8.1.4). Each record comprises a single command keyword, optionally followed by one or more parameters separated by spaces (2/0). All keywords are coded as text strings using lower-case alphabetic characters only (codes 6/1 to 7/10).

8.4.1 Command parameters

The following parameter types are used by the commands:

8.4.1.1 'hexadd' denotes a memory address coded in hexadecimal notation using the characters 0 - 9 and A - F (upper case) (codes 3/0 to 3/9 and 4/1 to 4/6). The relationship of this address to any terminal operating-system virtual memory addressing scheme is not specified by this protocol, but is expected to be the same as that used in the interpretation of load-address and execute-address (sections 4.3.2.10 and 4.3.2.11).

8.4.1.2 'filename' denotes a file in the terminal's filing system. The same recommendations as to usage apply as to the filename parameter specified in section 4.3.2.3. The following filenames are reserved:

- 'display'; output to be displayed
- 'printer'; output to be printed
- 'line'; reserved for future standardisation

8.4.1.3 'language' denotes a programming language; the syntax of this parameter is not specified by this protocol.

8.4.2 Commands

The following commands are available:

8.4.2.1 save filename [hexadd1 hexadd2]

Data from foreground memory is to be saved to file using the specified filename. If no address parameters are present, then the command refers to the program in foreground memory under the currently invoked language (eg BASIC). If address parameters are present, then the command refers to the data from hexadd1 to hexadd2-1 inclusive.

8.4.2.2 load filename [hexadd]

Data from a file with the specified filename is to be loaded into foreground memory. The memory address, if present, overrides any load address which may be stored with the file, or be defined by the local system.

8.4.2.3 exec filename [language]

Data in the file with the specified filename is to be executed; if the language parameter is present then the data is to be regarded as a program in that language, if no language parameter is present then the data is to be regarded as operating system commands in the current terminal operating system language.

8.4.2.4 run hexadd

The terminal's program pointer is to be set to the specified address.

8.4.2.5 rename filename1 filename2

The file with filename1 is to be renamed with filename2

8.4.2.6 copy filename1 filename2.

The file with filename1 is to be copied to create a new file with filename2.

8.4.2.7 del filename

The file with the specified filename is to be deleted from the terminal's filing system.

8.4.2.8 dir [filename]

The terminal's file directory is to be output to the specified filename (default is the terminal display). The format of the directory is not specified by this protocol.

8.4.2.9 quit

The videotex connection is to be abruptly terminated and the terminal returned to user control.

8.5 Coding of data intended for a standardised printer.

This section specifies the coding of data intended for the standardised printer (device '!P') in the auxiliary-device transfer application.

In this application, each transfer is intended to commence a new page or form, although the transfer may extend to more than one page or form of text. Each record is intended to commence a new row of text. Over-length records will be truncated.

8.5.1 Control codes.

The control codes specified for the virtual standardised printer are taken in general from the CEPT videotex display syntax or from ISO 6429. Terminals are expected to recognise the following control code syntaxes:

```
ESC [2/x ...] 3/x                    )
ESC [2/x ...] 4/x                    ) ESC = 1/11
ESC [2/x ...] 5/x                    )
ESC [2/x ...] 6/x                    )

CSI [3/x ...] [2/x ...] 4/x          ) CSI = 1/11 5/11
CSI [3/x ...] [2/x ...] 5/x          ) or
CSI [3/x ...] [2/x ...] 6/x          ) CSI = 9/11
```

where '[2/x ...]' denotes zero, one or more occurrences of a code 2/x

Any other single byte from columns 0, 1, 8 or 9.

Notes:
1 the code sequences ESC 4/x and ESC 5/x have the same meaning as the codes 8/x and 9/x respectively.
2 control codes and code sequences are non-spacing except for spacing videotex display attributes (section 8.5.1.10).

Terminals are expected to interpret and implement the following control codes:

8.5.1.1 Clear screen, CS. Coded 0/12 and with the meaning 'new page'.

8.5.1.2 Repeat, RPT. Coded 1/2, X and with the meaning 'repeat the last graphic character a further 'n' times, where n is the binary value of the six least significant bits of X.'

8.5.1.3 Active position address, APA. Coded 1/15, X', X, Y', Y (all parameters taken from columns 4 to 7 of the code table). This sequence causes the active position to be moved to the row specified by the bytes X' and X, and the column specified by the bytes Y' and Y, provided that the specified row number is equal to or higher than the current row number on the page. (The effect of attempting to overwrite text on the current row, or of addressing a column greater than that available on the printer, is not specified.) The row and column are specified as 12-bit values using the least significant 6 bits of 2 bytes, the first byte carrying the most significant bits. If both row and column address 0 are specified, then the active position should remain in the row and column in which it was situated before the APA sequence was received.

The operation of the following codes may be implementation-dependent:

8.5.1.4 Backspace, BS. Coded 0/8. The preferred action is to overprint character(s) previously received; the default action is to print only the second character received.

8.5.1.5 Unit separator, US. Coded 1/15, X, Y (X taken from columns 2 to 3 of the code table). This code sequence identifies the start of non-alphamosaic-coded data which may be ignored by the terminal unless it is within the specified repertoire of the terminal (eg DRCS definitions as required by the repertoire specification byte 4/5 in section 8.2.29.1). The end of the data string which may be ignored will be marked by a US or APA sequence.

8.5.1.6 Partial line down, PLD. Coded 9/11, 4/11. The following characters within the same record, or until the next CS, US or PLU character, are intended to be printed as subscripts; if PLU is already in effect then PLD has the effect only of cancelling the PLU.

8.5.1.7 Partial line up, PLU. Coded 9/11, 4/12. The following characters within the same record, or until the next CS, US or PLD character, are intended to be printed as superscripts; if PLD is already in effect then PLU has the effect only of cancelling the PLD.

8.5.1.8 Select graphic rendition, SGR. Coded 9/11, [P1, [3/11, P2, [3/11, P3, [3/11, P4,]]]] 6/13 where each parameter, Pn, may take one of the values:

- 3/1 bold or increased intensity
- 3/3 italic
- 3/4 underlined
- 3/9 crossed-out

Each parameter affects the representation of subsequent text up to the next SGR sequence, or to the end of the transfer if no further SGR is encountered.

8.5.1.9 Graphic size modification, GSM. Coded 9/11, P1, 3/11, P2, 2/0, 4/2 where the parameters P1 and P2 are each encoded as decimal numbers using the bytes 3/0 to 3/9, specifying the required height and width respectively of subsequent characters as percentages of the default height and width. Each parameter affects the representation of subsequent text up to the next GSM sequence, or to the end of the transfer if no further GSM is encountered.

8.5.1.10 Videotex display attributes. Coded 8/x or 9/x (or alternatively ESC 4/x or ESC 5/x). These attributes may be used to indicate a preferred foreground or background colour or character size for the printed text. (Use of a videotex display attribute to change character size is to be regarded as an alternative to the use of GSM, it does not alter the default height referenced by the GSM command. For example, the code 8/13 (double height) has an identical effect in the terminal to the code sequence 9/11, 3/2, 3/0, 3/0, 3/11, 3/1, 3/0, 3/0, 2/0, 4/2.)

8.5.2 Character codes

Graphic characters will be encoded according to the CEPT videotex display syntax.

8.6 Coding of Responses by the Terminal

8.6.1 T-Association reject. This response is coded as the single character 3/9 ('9') unless it has been redefined as specified in 8.2.2.

8.6.2 T-Read-Restart. This response is coded as the single character 3/7 ('7').

8.6.3 T-Application-Reject. This response is coded as the single character 3/6 ('6').

ANNEX A - USE OF CHECKSUMS

This Annex describes the additional provisions which apply if it is necessary to include checksums in the telesoftware VPDE's.

A checksum (BCS) follows every D-End-Group if the 'define checksum use' parameter is set in the initial D-Set-Mode DDU.

A.1 Use of Dialogue Data Units

Operation is as in section 5 with the following additional provisions.

A.1.1 The terminal should not receive a total of more than 2047 bytes of data (as received from line but excluding stuffed US characters in the case of mode 1) from the start of an initial VPDE, before a terminal response is required.

A.1.2 Received DDU's and TDU's should not be acted on until a D-End group is received, and the checksum and all sequence codes are correct.

A.1.3 If an alphamosaic delimiter (US x y) is received during reception of a VPDE, this is an error (ie, with reference to Figure 1, alphamosaic data may only follow a D-End-Group).

A.2 Coding of checksum use parameter.

This parameter is coded in the same byte as the 'define mode' parameter. If BCS are to be used, then this byte is coded from column 3 of the code table instead of column 4 as specified in section 7.2.1.1. The use of BCS cannot be changed by the use of D-Control, however, so the use of column 3 or 4 is to be ignored when the mode parameter is included in a D-Control DDU.

A.3 Calculation of Block Check Sequence

If the value of the Mode parameter indicates that a BCS will be transmitted, then a checksum is calculated as follows and encoded in the three bytes immediately following D-Flags.

The checksum is initialised:
- after receipt of the first delimiter of a D-Set mode DDU, and
- after receipt of the first delimiter after a D-End group DDU.

The checksum is then accumulated over all bytes up to the D-Flags byte immediately preceding the checksum (see Figure A.1).

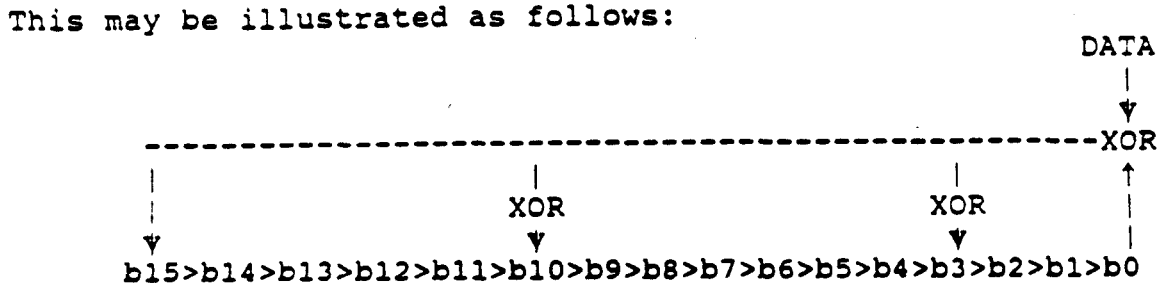
The checksum is the 16-bit frame checking sequence specified in CCITT Rec. X.25, formed by division by the polynomial

$$x^{16} + x^{12} + x^5 + 1$$

This may be calculated according to the following procedure. The checksum, b15.....b0, is initialised to all 1's. It is then modified in turn for each bit of the data. For this purpose the data is treated as 8-bit bytes, and the least significant bits in each byte are taken first; if the most significant bit of each data byte is a parity bit it is set to zero before the checksum calculation. The modification consists of:

a) an exclusive-OR of b0 with the next data bit,
b) a one-bit shift in which a zero bit is brought into b15 of the checksum,
c) an exclusive-OR of the result of (a) with the new bits b15, b10 and b3 of the checksum.

This may be illustrated as follows:



The checksum is finally complemented (1's complement) and mapped into three bytes as described in section 1.2 and illustrated in Figure 2b.

A.2.1 Example of CRC Calculation

Figure A.2 illustrates the calculation of the CRC checksum for the D-Set mode DDU (mode 0):

1/15, 3/14, 2/7, 4/0, 4/0, 1/15, 3/14, 3/0, . . .

(Note: the first two bytes are not included in the checksum)

The sequences which may be transmitted are therefore:

1/15, 3/14, 2/7, 4/0, 4/0, 1/15, 3/14, 3/0, 7/4, 4/8, 6/11

or with even parity:

9/15, 3/14, 2/7, 12/0, 12/0, 9/15, 3/14, 3/0, 7/4, 4/8, 14/11

FIGURE A.1
EXAMPLES OF DDU CODING WITH BCS (compare Fig 4 of main text)

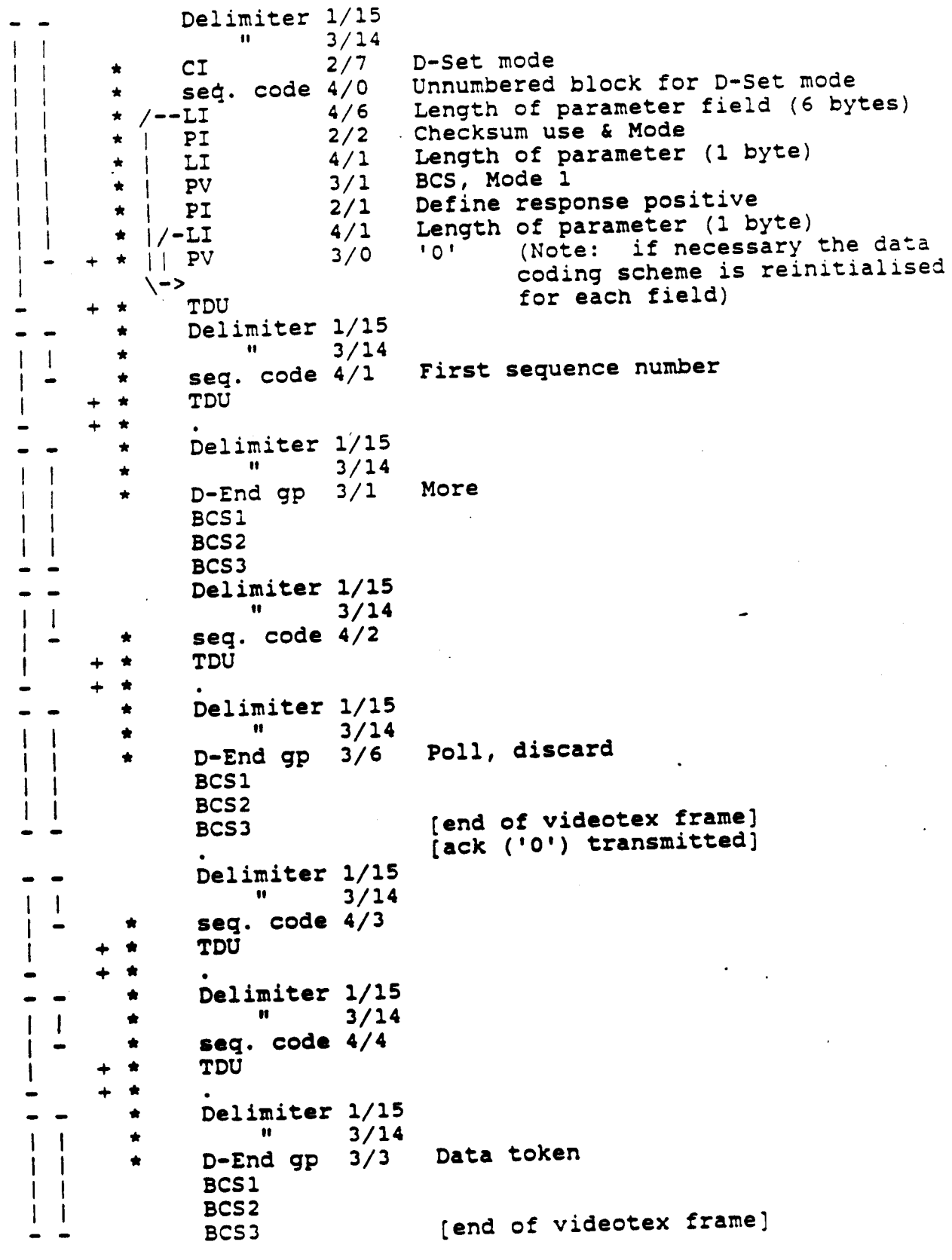


FIGURE A.2 - CRC CALCULATION

	bit15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Next data bit	
Initial value:	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2
	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	/
	1	0	0	0	1	0	1	1	1	1	1	1	0	1	1	1	0	7
	1	1	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1	
	0	1	1	0	0	0	0	0	1	1	1	1	1	0	0	1	0	
	1	0	1	1	0	1	0	0	0	1	1	1	0	1	0	0	0	
	0	1	0	1	1	0	1	0	0	0	1	1	1	0	1	0	0	
	0	0	1	0	1	1	0	1	0	0	0	1	1	1	0	1	0	
	1	0	0	1	0	0	1	0	1	0	0	0	0	1	1	0	0	4
	0	1	0	0	1	0	0	1	0	1	0	0	0	0	1	1	0	/
	1	0	1	0	0	0	0	0	1	0	1	0	1	0	0	1	0	0
	1	1	0	1	0	1	0	0	0	1	0	1	1	1	0	0	0	
	0	1	1	0	1	0	1	0	0	0	1	0	1	1	1	0	1	
	1	0	1	1	0	0	0	1	0	0	0	1	1	1	1	1	0	
	1	1	0	1	1	1	0	0	1	0	0	0	0	1	1	1	0	
	1	1	1	0	1	0	1	0	0	1	0	0	1	0	1	1	0	
	1	1	1	1	0	0	0	1	0	0	1	0	1	1	0	1	0	4
	1	1	1	1	1	1	0	0	1	0	0	1	1	1	1	0	0	/
	0	1	1	1	1	1	1	0	0	1	0	0	1	1	1	1	0	0
	1	0	1	1	1	0	1	1	0	0	1	0	1	1	1	1	0	
	1	1	0	1	1	0	0	1	1	0	0	1	1	1	1	1	1	
	0	1	1	0	1	1	0	0	1	1	0	0	1	1	1	1	0	
	1	0	1	1	0	0	1	0	0	1	1	0	1	1	1	1	1	
	0	1	0	1	1	0	0	1	0	0	1	1	0	1	1	1	1	
	0	0	1	0	1	1	0	0	1	0	0	1	1	0	1	1	1	1
	0	0	0	1	0	1	1	0	0	1	0	0	1	1	0	1	1	/
	0	0	0	0	1	0	1	1	0	0	1	0	0	1	1	0	1	1
	1	0	0	0	0	0	0	1	1	0	0	1	1	0	1	1	0	5
	1	1	0	0	0	1	0	0	1	1	0	0	0	1	0	1	0	
	1	1	1	0	0	1	1	0	0	1	1	0	1	0	1	0	0	
	0	1	1	1	0	0	1	1	0	0	1	1	0	1	0	1	0	
	1	0	1	1	1	1	0	1	1	0	0	1	0	0	1	0	1	
	1	1	0	1	1	0	1	0	1	1	0	0	0	0	0	1	1	3
	0	1	1	0	1	1	0	1	0	1	1	0	0	0	0	0	1	/
	1	0	1	1	0	0	1	0	1	0	1	1	1	0	0	0	1	1
	1	1	0	1	1	1	0	1	0	1	0	1	0	1	0	0	1	4
	1	1	1	0	1	0	1	0	1	0	1	0	0	0	1	0	0	
	0	1	1	1	0	1	0	1	0	1	0	1	0	0	0	1	0	
	1	0	1	1	1	1	1	0	1	0	1	0	0	0	0	0	0	
	0	1	0	1	1	1	1	1	0	1	0	1	0	0	0	0	0	
	0	0	1	0	1	1	1	1	1	0	1	0	1	0	0	0	0	3
	0	0	0	1	0	1	1	1	1	1	0	1	0	1	0	0	0	/
	0	0	0	0	1	0	1	1	1	1	1	0	1	0	1	0	1	0
	1	0	0	0	0	0	0	1	1	1	1	1	1	1	0	1	1	
	0	1	0	0	0	0	0	0	1	1	1	1	1	1	1	0	0	
	0	0	1	0	0	0	0	0	0	1	1	1	1	1	1	1	0	
Final val:	1	0	0	1	0	1	0	0	0	0	1	1	0	1	1	1		
Complement:	0	1	1	0	1	0	1	1	1	1	0	0	1	0	0	0		

These 16 bits map for transmission into:

1st byte	X	1	1	1	0	1	0	0
2nd byte	X	1	0	0	1	0	0	0
3rd byte	X	1	1	0	1	0	1	1

ANNEX B - EXAMPLES OF PROCESSABLE DATA APPLICATIONS

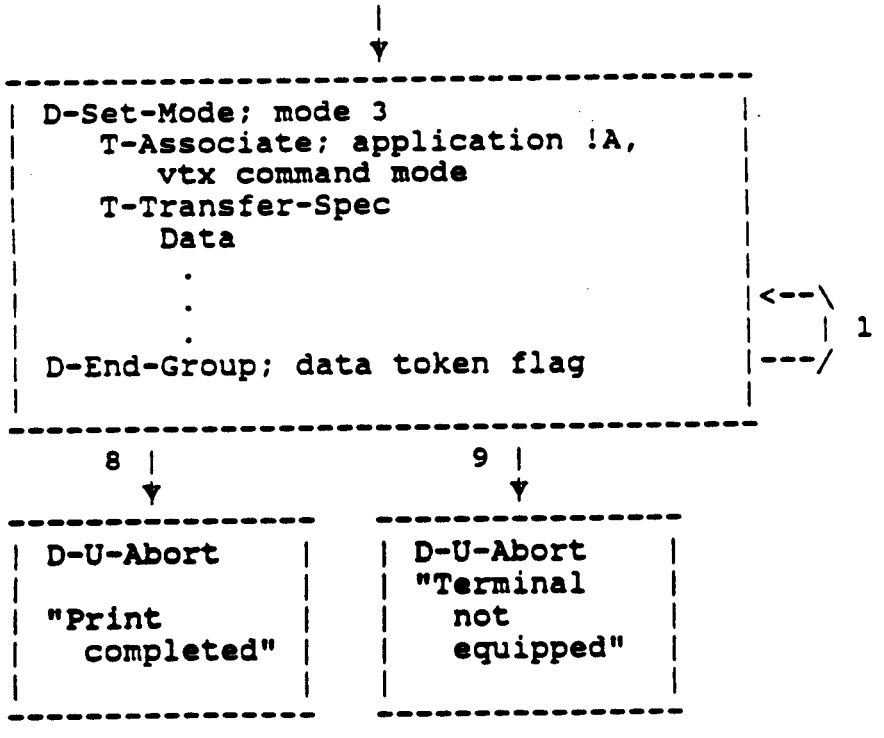
The following examples illustrate the use of videotex processable data facilities for a range of applications. The examples are presented assuming a fixed, tree-structured videotex database, but the same sequences could, of course, be implemented interactively if required.

In the following examples, each box indicates a videotex 'frame', and the arrows indicate the effect of videotex routes or commands. The text within the boxes indicates the DDU's and TDU's needed to effect the required task; text in " " indicates possible messages to the user.

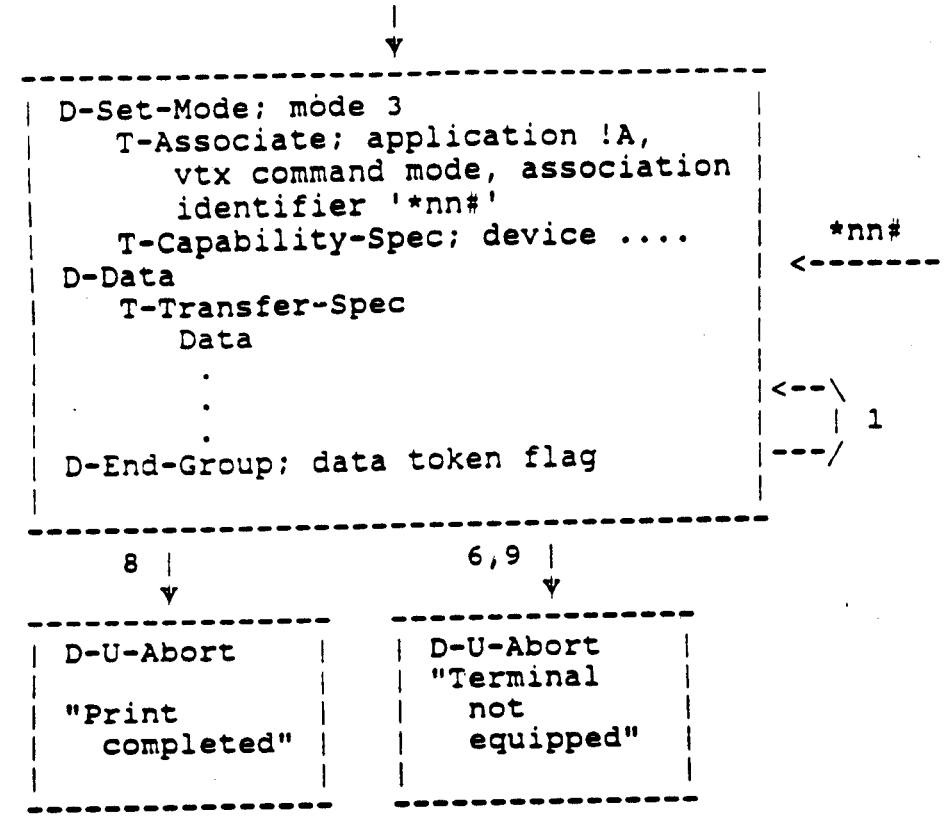
Examples of the coding of protocol sequences are given in examples 6 and 7.

Example 1. Print a short document, minimum overhead

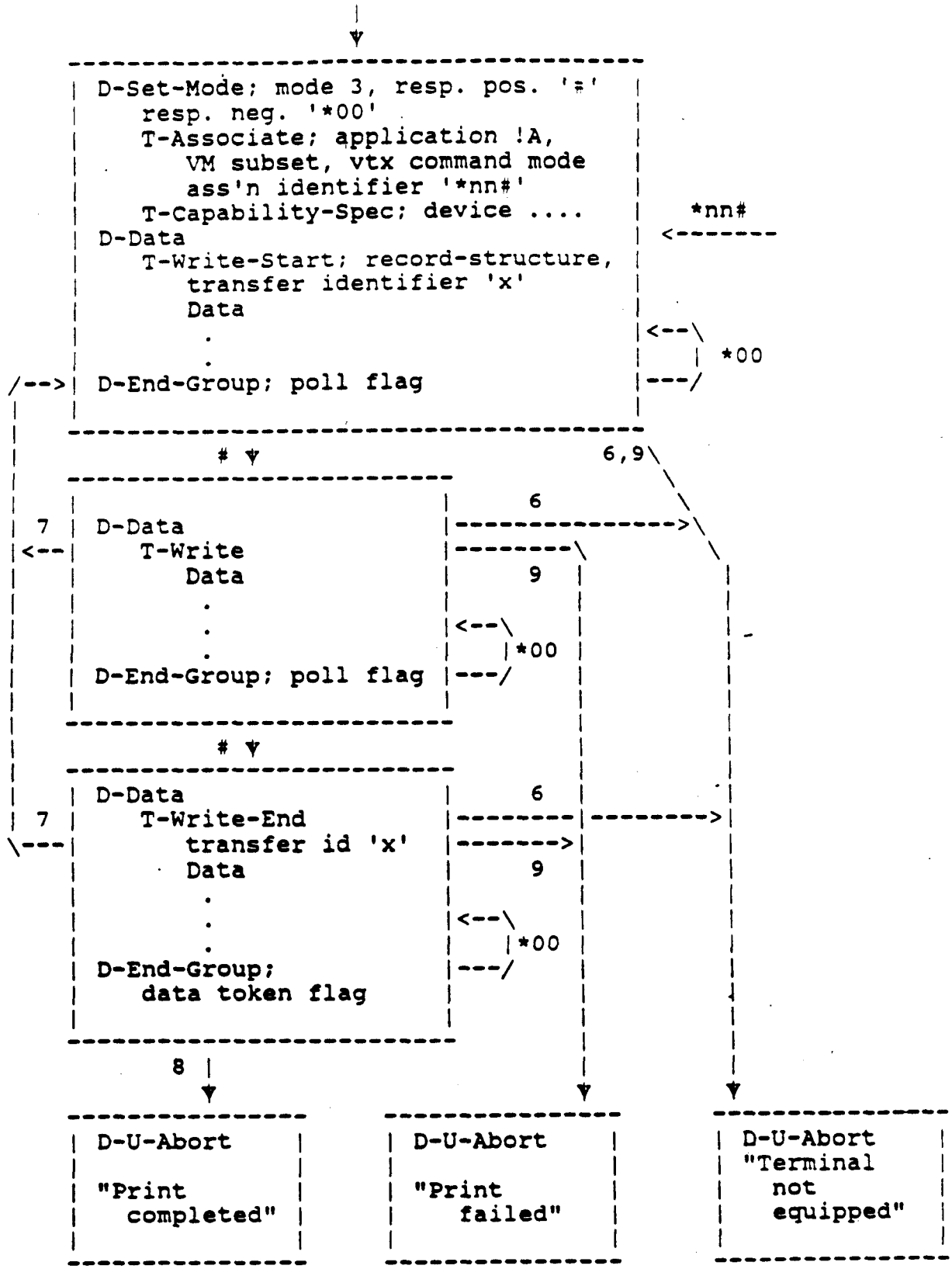
(Auxiliary device understood to be printer by means outside the protocol.)



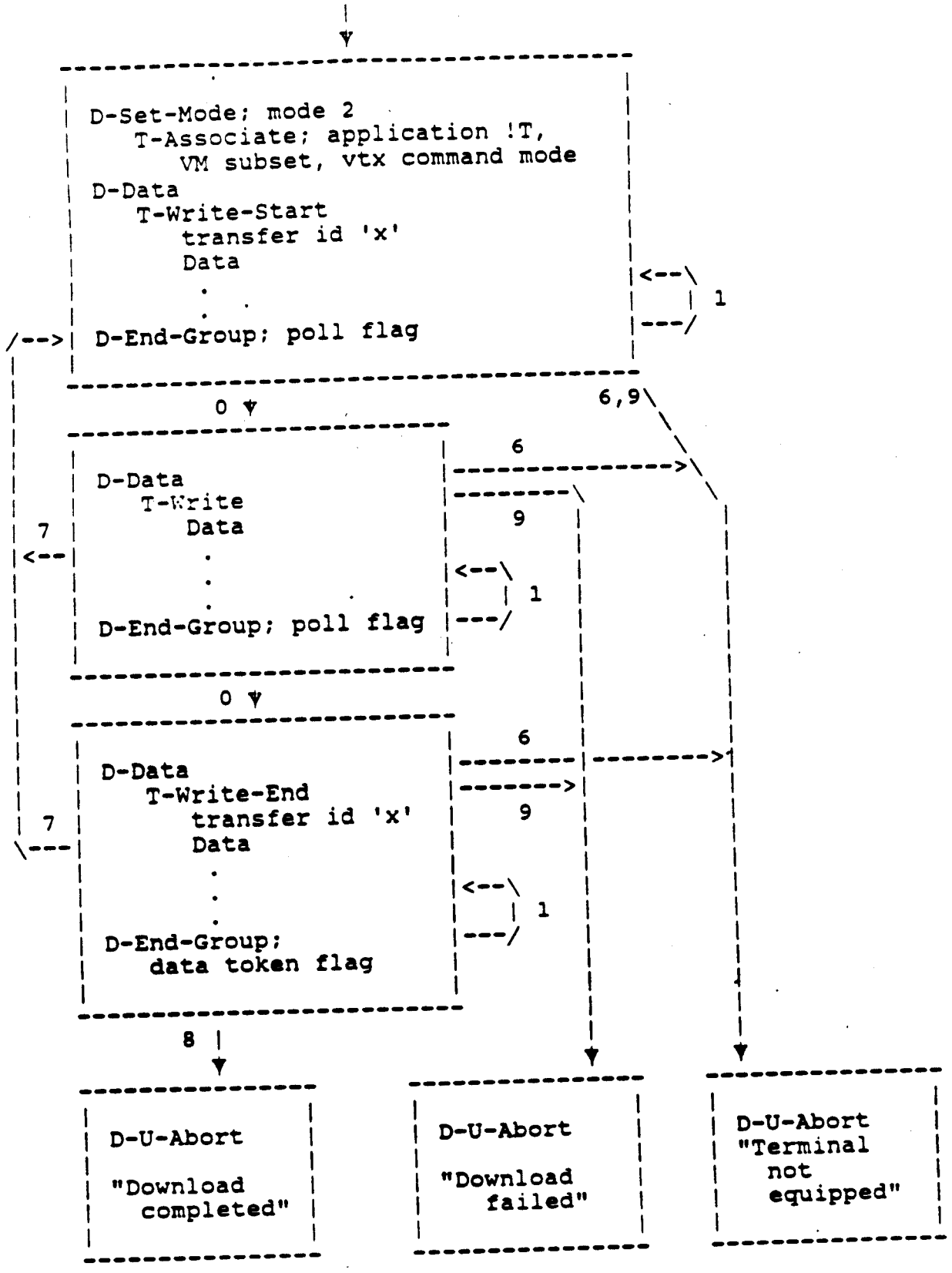
Example 2.
Print a short document, using capability spec. and auto recovery



Example 3.
Print a long document, using capability spec. and auto recovery

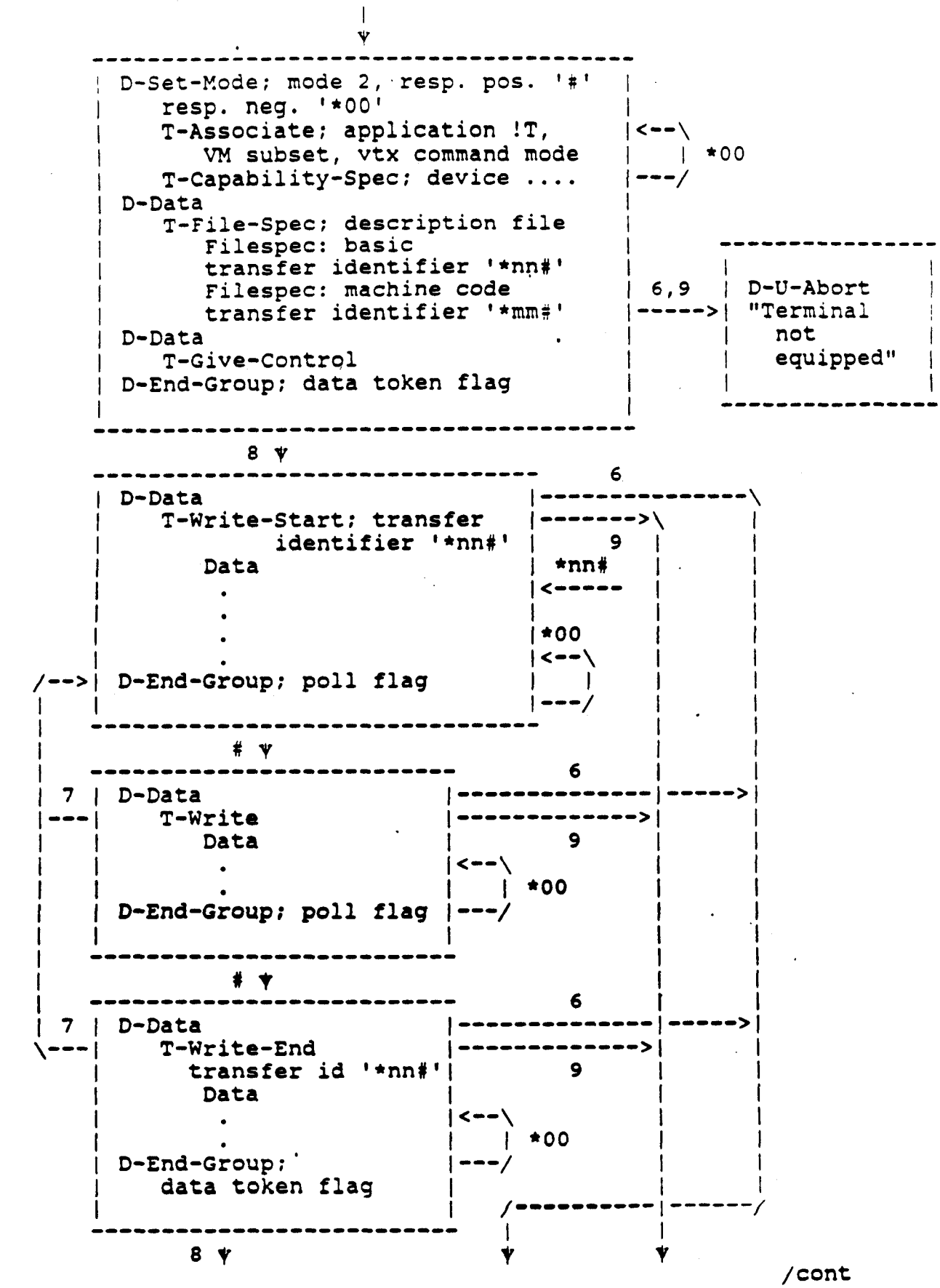


Example 4.
Download one telesoftware file, minimum overhead



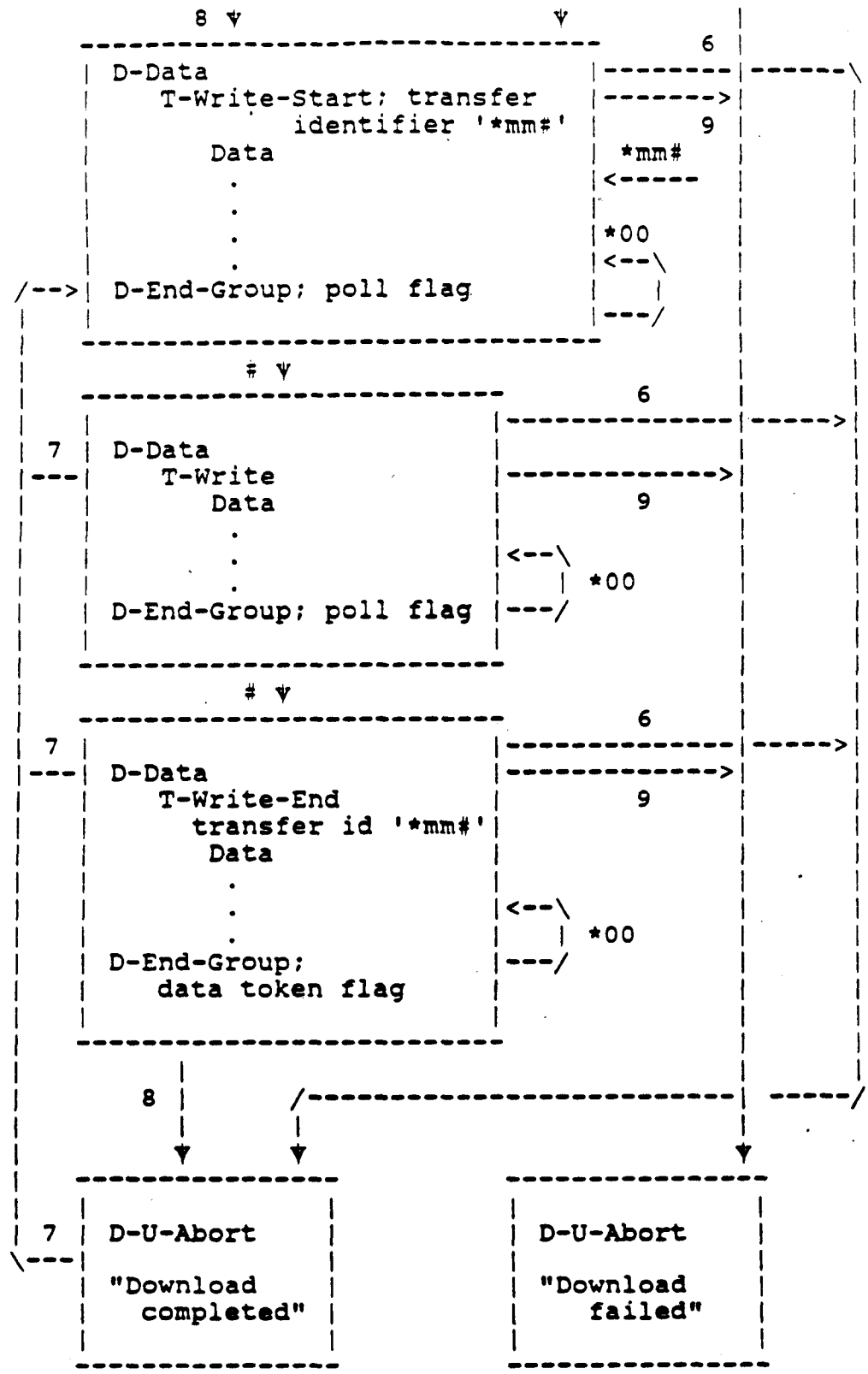
Example 5.

Download two telesoftware files, using capability spec. and auto recovery



/cont

5. (cont)


8
D-Data
T-Write-Start; transfer identifier '*mm#'
Data
D-End-Group; poll flag
6
9
*mm#
*00
#
D-Data
T-Write
Data
D-End-Group; poll flag
7
6
9
*00
#
D-Data
T-Write-End transfer id '*mm#'
Data
D-End-Group; data token flag
7
6
9
*00
8
7
D-U-Abort
"Download completed"
D-U-Abort
"Download failed"

Example 6. The protocol information relating to example 1 above would be coded as follows. Each line of text corresponds to one byte as transmitted to line.

Line Data			TDU Data		
Delimiter	1/15				
"	3/14				
CI	2/7	D-Set mode			
seq. code	4/0				
/--LI	4/3				
| PI	2/2	Checksum use & Mode			
|/-LI	4/1				
|| PV	4/3	No BCS, Mode 3			
\->					
TDU	2/3		CI	2/3	T-Associate
|	7/14				
|	5/8		/--LI	0/8	
|	3/1		| Str	3/1	
|	4/5		| PI	4/5	Application-name
|	7/14		|		
|	5/2		|/-LI	0/2	
|	2/1		|| PV	2/1	) Auxiliary-device
|	4/1		|| PV	4/1	)
|			|\>		
|	4/0		| PI	4/0	Terminal flags
|	7/14		|		
|	5/1		|/-LI	0/1	
|	4/2		|| PV	4/2	Command mode
-			\->		
TDU	6/3		CI	6/3	T-Transfer-Spec
|	7/14				
|	5/1		/--LI	0/1	
|	3/1		| Str	3/1	
|			\->		
|	.		Data		
|	.		.		
|	.		.		
|	.		.		
-	.		.		
Delimiter	1/15				
"	3/14				
D-End gp	3/3	Data token [end of videotex frame]			
Delimiter	1/15				
"	3/14				
CI	2/9	D-U-Abort			
seq. code	4/1				
/--LI	4/0				
\->					

Example 7. The protocol information relating to the first part of example 5 above would be coded as follows.

```
   Delimiter 1/15
       "     3/14
   CI        2/7.  D-Set mode
   seq. code 4/0
/--LI        4/13
|  PI        2/2   Checksum use & Mode
|/-LI        4/1
|| PV        4/2   No BCS, Mode 2
|\>
|  PI        2/1   Define D-response positive
|/-LI        4/2
|| PV        5/0   )
|| PV        5/15  ) 5/15 #
|\>
|  PI        2/5   Define D-response negative
|/-LI        4/4
|| PV        4/0   )
|| PV        6/10  ) 2/10 *
|| PV        7/0   ) 3/0  0
|| PV        7/0   ) 3/0  0
\->             ..............
   TDU       4/0
    |        6/3                      CI  2/3   T-Associate
    |        4/11                  /--LI  0/11
    |        7/1 .............. |   Str 3/1
    |        5/0                 |
    |        4/5                 |   PI  4/5   Application-name
    |        4/2                 |/-LI  0/2
    |        6/1 .............. ||  PV  2/1   ) Telesoftware
    |        5/4                 ||  PV  5/4   )
    |        5/4                 |\>
    |        4/4                 |   PI  4/4   Optional subset
    |        4/1 .............. |/-LI  0/1
    |        5/4                 ||  PV  4/1   VM
    |        4/1                 |\>
    |        4/0                 |   PI  4/0   Terminal flags
    |        4/1 .............. |/-LI  0/1
    |        5/4                 ||
    |        4/2                 ||  PV  4/2   Command mode
    -                            \->
   TDU       6/1                      CI  6/1   T-Capability-Spec
    |        4/6 .............. /--LI  0/6
    |        4/4                 |
    |        7/1                 |   Str 3/1
    |        6/1                 |   PI  6/1   Target machine
    |        4/3 .............. |/-LI  0/3
    |        5/5                 ||
    |        5/8                 ||  PV  5/8   X
    |        5/9                 ||  PV  5/9   Y
    -        5/10.............. ||  PV  5/10  Z
   Delimiter 1/15               \->
       "     3/14
   seq. code 4/1   First sequence number
   TDU         .
    |          .
    .          .
```

History

Document history	
November 1990	First Edition
March 1996	Converted into Adobe Acrobat Portable Document Format (PDF)