INTERNATIONAL TELECOMMUNICATION UNION

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

X.37

(04/95)

DATA NETWORKS AND OPEN SYSTEM COMMUNICATIONS

PUBLIC DATA NETWORKS – INTERFACES

ENCAPSULATION IN X.25 PACKETS OF VARIOUS PROTOCOLS INCLUDING FRAME RELAY

ITU-T Recommendation X.37

(Previously "CCITT Recommendation")

FOREWORD

The ITU-T (Telecommunication Standardization Sector) is a permanent organ of the International Telecommunication Union (ITU). The ITU-T is responsible for studying technical, operating and tariff questions and issuing Recommendations on them with a view to standardizing telecommunications on a worldwide basis.

The World Telecommunication Standardization Conference (WTSC), which meets every four years, establishes the topics for study by the ITU-T Study Groups which, in their turn, produce Recommendations on these topics.

The approval of Recommendations by the Members of the ITU-T is covered by the procedure laid down in WTSC Resolution No. 1 (Helsinki, March 1-12, 1993).

ITU-T Recommendation X.37 was prepared by ITU-T Study Group 7 (1993-1996) and was approved under the WTSC Resolution No. 1 procedure on the 10th of April 1995.

____

NOTE

In this Recommendation, the expression "Administration" is used for conciseness to indicate both a telecommunication administration and a recognized operating agency.

ITU-X SERIES RECOMMENDATIONS

DATA NETWORKS AND OPEN SYSTEM COMMUNICATIONS

(February 1994)

ORGANIZATION OF X-SERIES RECOMMENDATIONS

Subject area	Recommendation Series
PUBLIC DATA NETWORKS	
Services and Facilities	X.1-X.19
Interfaces	X.20-X.49
Transmission, Signalling and Switching	X.50-X.89
Network Aspects	X.90-X.149
Maintenance	X.150-X.179
Administrative Arrangements	X.180-X.199
OPEN SYSTEMS INTERCONNECTION	
Model and Notation	X.200-X.209
Service Definitions	X.210-X.219
Connection-mode Protocol Specifications	X.220-X.229
Connectionless-mode Protocol Specifications	X.230-X.239
PICS Proformas	X.240-X.259
Protocol Identification	X.260-X.269
Security Protocols	X.270-X.279
Layer Managed Objects	X.280-X.289
Conformance Testing	X.290-X.299
INTERWORKING BETWEEN NETWORKS	
General	X.300-X.349
Mobile Data Transmission Systems	X.350-X.369
Management	X.370-X.399
MESSAGE HANDLING SYSTEMS	X.400-X.499
DIRECTORY	X.500-X.599
OSI NETWORKING AND SYSTEM ASPECTS	
Networking	X.600-X.649
Naming, Addressing and Registration	X.650-X.679
Abstract Syntax Notation One (ASN.1)	X.680-X.699
OSI MANAGEMENT	X.700-X.799
SECURITY	X.800-X.849
OSI APPLICATIONS	
Commitment, Concurrency and Recovery	X.850-X.859
Transaction Processing	X.860-X.879
Remote Operations	X.880-X.899
OPEN DISTRIBUTED PROCESSING	X.900-X.999

CONTENTS

Page

SUMMARY

This Recommendation specifies the X.25 call set-up and call clearing procedures used by an encapsulation function and the data transfer phase which follows, when using Packet Switched Data Transmission Service to transmit Protocol Data Units of various protocols. In addition to general procedures, this Recommendation deals with the specific encapsulation in X.25 packets of Frame Relay frames and IP datagrams.

ENCAPSULATION IN X.25 PACKETS OF VARIOUS PROTOCOLS INCLUDING FRAME RELAY

(Geneva, 1995)

1 Scope

This Recommendation defines the use of Packet Switched Data Transmission Service to transmit Protocol Data Units of various protocols including:

- protocols used in Local Area Network;
- Frame Relay.

The aim of this Recommendation is to favour:

- inter-operability between equipments (e.g. routers);
- interworking between DTEs which are connected to PDNs but do not use the same data transmission services (e.g. Frame Relay and Packet Switched data transmission service);
- network optimisation, both on user and operator levels, by increasing the use of existing and forthcoming infrastructures;
- promotion of Packet Switched Data Transmission Services;
- promotion of X.75 use for PDN interconnection;

This Recommendation makes use of the protocol identifiers defined in ISO/IEC TR 9577.

Appendix I provides application examples which illustrate some of the preceding items.

2 References

The following Recommendations, and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision: all users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published.

- ITU-T Recommendation X.25 (1993), *Interface between Data Terminal Equipment (DTE) and Data Circuit-Terminating Equipment (DCE) for terminals operating in the packet mode and connected to public data networks by dedicated circuits.*
- ITU-T Recommendation X.29 (1993), *Procedures for the exchange of control information and user data between a Packet Assembly/Disassembly (PAD) facility and a packet mode DTE or another PAD.*
- ITU-T Recommendation X.36 (1995), *Interface between Data Terminal Equipment (DTE) and Data Circuit-terminating Equipment (DCE) for public data networks providing frame relay data transmission service by dedicated circuit.*
- CCITT Recommendation X.39 (1992), *Procedures for the exchange of control information and user data between a Facsimile Packet Assembly/Disassembly (FPAD) facility and a packet mode Data Terminal Equipment (DTE) or another FPAD.*
- ITU-T Recommendation X.75 (1993), *Packet switched signalling system between public networks providing data transmission services.*
- ITU-T Recommendation X.76 (1995), *Network-to-network interface between public data networks providing the frame relay data transmission service.*

– ITU-T Recommendation X.224 (1993), *Protocol for providing the OSI connection-mode transport service.*

– ISO/IEC 8073:1992, *Information technology – Telecommunications and information exchange between systems – Open Systems Interconnection – Protocol for providing the connection mode network service.*

– ITU-T Recommendation X.622 (1994) | ISO/IEC 8473-3 ...[1], *Information technology – Potocol for providing the connectionless-mode Network service: Povision of the underlying service by an X.25 subnetwork.*

– ISO/IEC TR 9577 (second edition), *Infomation technology – Telecommunications and information exchange between systems – Protocol identification in the network layer.*

3 Terms and definitions

For the purpose of this Recommendation, the following definitions apply:

3.1 encapsulation function: A function which may reside inside an X.25 DTE or inside a Packet Switched Public Data Network. When it receives a PDU of a given protocol from the user equipment, it transmits this PDU within a complete packet sequence on an X.25 virtual call. When it receives a PDU of a given protocol within a complete packet sequence, it retrieves this PDU and transmits it towards the user equipment. It may also provide interworking function between the local interface protocol and Recommendation X.25.

3.2 local interface: The interface which is used to transfer Protocol Data Units of one protocol (or more) between an encapsulation function and a user equipment, before encapsulation and after de-encapsulation in X.25 packets.

3.3 PDU: The abbreviation PDU refers to Protocol Data Units of the protocol(s) which is(are) encapsulated in X.25 packets.

4 Abbreviations

For the purpose of this Recommendation the following abbreviations are used:

BECN	Backward Explicit Congestion Notification
CLNP	Connectionless Network Protocol
C/R	Command/Response
CRC	Cyclic Redundancy Code
DCE	Data Circuit-terminating Equipment
DE	Discard Eligibility
DLCI	Data Link Connection Identifier
DNIC	Data Network Identification Code
DTE	Data Terminal Equipment
IP	Internet Protocol
FECN	Forward Explicit Congestion Notification
FR	Frame Relay
LAN	Local Area Network
LCN	Logical Channel Number
MTU	Maximum Transmission Unit
PICS	Protocol Implementation Conformance Statement
PDU	Protocol Data Unit

[1] To be published.

PSPDN	Packet Switched Public Data Network
PVC	Permanent Virtual Circuit (for Frame Relay)
RFC	Request for Comment
SNAP	Single Network Access Protocol

5 General

This Recommendation defines the X.25 call set-up and call clearing procedures used by an encapsulation function, and the data transfer phase which follows.

These procedures are categorised as follows:

– procedures which are independent of the protocol being encapsulated: see clauses 6, 7 and 8;

– procedures which are dependent on the protocol being encapsulated: see clauses 9 and 10.

When referring to Recommendation X.25 in this Recommendation, a reference is made to all uses of the packet layer which includes cases described in Recommendations X.31, X.32, etc.

When referring to Recommendation X.36, a reference is made to all uses of Frame Relay protocol which includes cases described in ISDN Recommendations (See Q- and I-Series Recommendations).

When dealing with encapsulation of various protocols, a given encapsulated protocol is identified by an identifier defined in ISO/IEC TR 9577 (second edition). In relation to X.25, relevant material is given on the use of the following ISO/IEC TR 9577 (second edition) defined protocol identifiers:

– multi-protocol encapsulation (hexadecimal 00), see 5.2.1.2, 6.3.4.1 and 8.2.2;

– IEEE SNAP convention (hexadecimal 80), see 6.3.4.1;

– identification of the protocol through other means (hexadecimal A0) see 6.3.4.1;

– Frame Relay according to X.37 (hexadecimal A8) see clause 9;

– Internet Protocol (hexadecimal CC) see clause 10.

NOTE –At the time of publication, code points hexadecimal A0 and A8 are agreed to be included in the third version of ISO/IEC TR 9577 and in forthcoming equivalent ITU-T Recommendations.

In addition to ISO/IEC TR 9577 aspects for X.25, Appendix III provides additional protocol identification mechanisms for data packets when the X.25 Q-bit is used.

In addition to this Recommendation, relevant material can be found in:

– Recommendation X.622 | ISO/IEC 8473-3, encapsulation of CLNP;

– Recommendation X.39, encapsulation of data from a Fax PAD;

– Recommendation X.29, encapsulation of data from a start stop PAD.

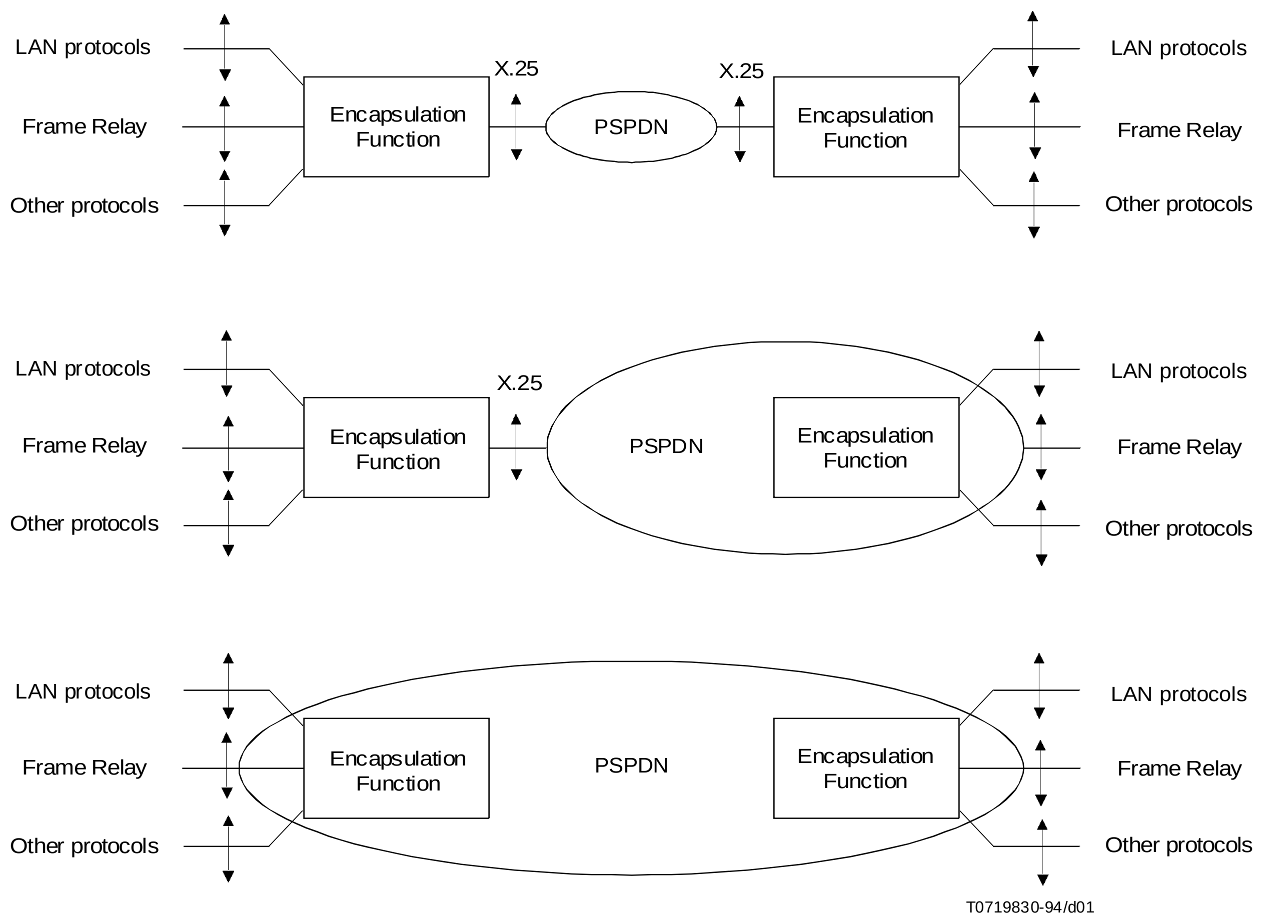
To claim its compliance to this Recommendation, a network or an encapsulation function must follow the requirements defined in Annex A.

5.1 Encapsulation function location

As shown in Figure 1, the encapsulation function may reside inside an X.25 DTE or inside the PSPDN.

5.2 Basic features

This subclause describes the basic features regarding the way protocols are encapsulated and transferred over an X.25 virtual call between two encapsulation functions.



NOTES

1 The protocols indicated here on the local interface are only examples.

2 The above depiction may imply that a system containing an encapsulation function does not originate any encapsulated protocols. This may not be the case as shown in Appendix I.

FIGURE 1/X.37

Possible locations of the encapsulation function

5.2.1 Encapsulation methods

The encapsulation method is negotiated at call set-up between the two encapsulation functions.

5.2.1.1 Mono-protocol encapsulation

The mono-protocol method allows to encapsulate PDUs of a single protocol over an X.25 virtual call.

Both encapsulation functions identify the encapsulated protocol at call set-up in the call user data field of the call set-up packets using protocol identifiers which are defined in ISO/IEC TR 9577 (second edition).

5.2.1.2 Multi-protocol encapsulation

The multi-protocol method allows to encapsulate PDUs of several protocols over a single X.25 virtual call.

Both encapsulation functions identify the multi-protocol encapsulation method at call set-up using the call user data field of the call set-up packets.

Each PDU to be transmitted on the X.25 virtual call is preceded by a protocol identifier which is defined in ISO/IEC TR 9577 (second edition).

5.2.2 Transfer methods

The transfer method is negotiated at call set-up between the two encapsulation functions.

5.2.2.1 Mono-PDU transfer

In case of mono-protocol encapsulation, the mono-PDU transfer allows the transfer of a single PDU per complete packet sequence. In the case of multi-protocol encapsulation, the mono-PDU transfer allows the transfer of a single PDU plus its protocol identifier per complete packet sequence.

Both encapsulation functions identify the mono-PDU transfer at call set-up using the call user data field of the call set-up packets.

5.2.2.2 Multi-PDU transfer

The multi-PDU transfer allows to transfer, per complete packet sequence, more than one PDU. Before each PDU (plus its protocol identifier in case of multi-protocol encapsulation) a delimiter is added.

Both encapsulation functions identify the multi-PDU transfer and the format of the delimiter at call set-up using the call user data field of the call set-up packets.

5.3 Use of X.75 for PDN interconnection

Since

- – the procedures used by an encapsulation function regardless of its location are consistent with Recommendation X.25;
- – the negotiation of the working mode of the encapsulation functions is made at call set-up using call user data fields;
- – transfer of encapsulated PDUs is achieved through X.25 data packets,

the use of X.75 signalling system between PDNs enables X.25 virtual call to be set-up and used between encapsulation functions connected to the different involved PDNs.

By using encapsulation functions, Recommendation X.75 does carry traffic of other Data Transmission Services such as FR or IP encapsulated in Recommendation X.25 as shown in Figure 2.

This promotion of X.75 use does not preclude the definition neither the use of other Recommendations for PDNs interconnection (e.g. Recommendations X.76 “network-to-network interface between public data networks providing frame relay data transmission services”).

5.4 Protocol stack on the local interface

This subclause gives information about how an encapsulation function may handle the dynamic negotiation of the encapsulated protocol according to the protocol stack used on the local interface. It does not describe the protocols which are used on the local interface and which are encapsulated on the X.25 virtual call, but an encapsulation function should have sufficient knowledge of the local protocol stack to send and receive PDU from the local interface. The encapsulation function should also know which protocol needs to be encapsulated. This information is maintained by means not described in this Recommendation.

In the Figure 3 example, the encapsulation function knows that the protocol Pk needs to be encapsulated and is able to terminate all the protocols between protocol Pk and the physical layer.

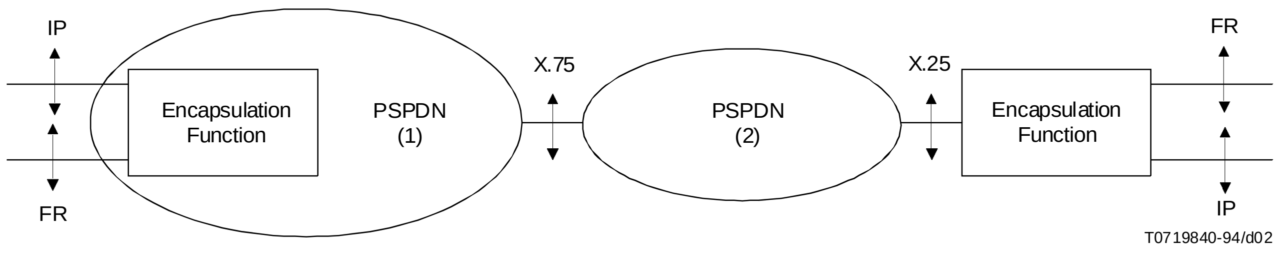
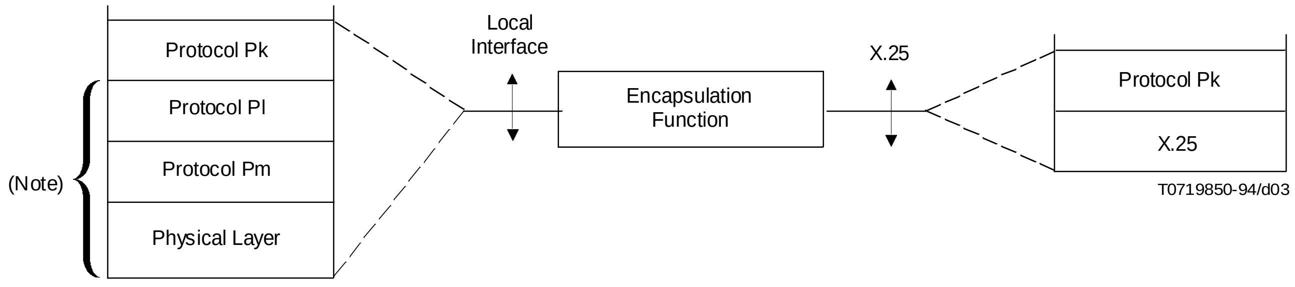


FIGURE 2/X.37

Transmission of FR frames or IP datagrams between PDNs using X.75 signalling system



NOTE – These protocols are terminated by the encapsulation function.

FIGURE 3/X.37

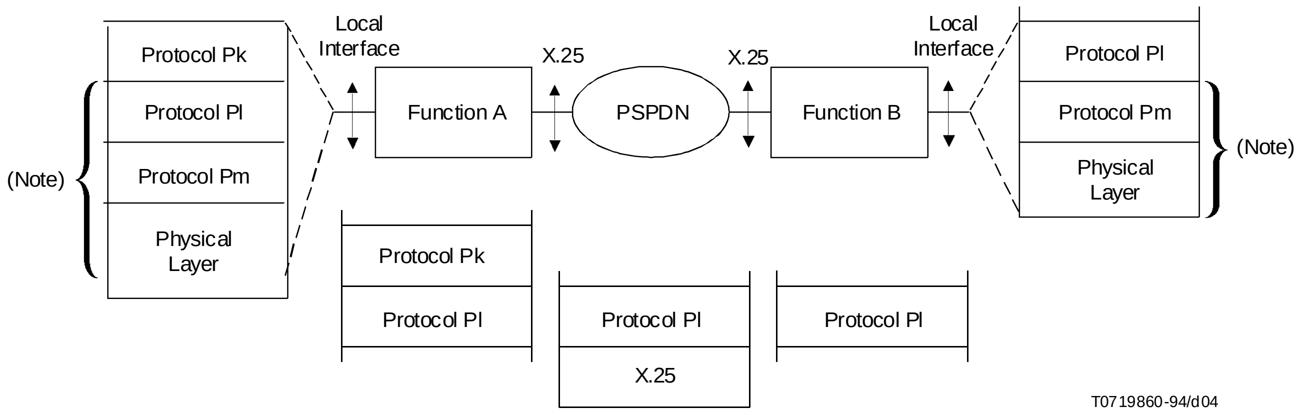
Knowledge of the local protocol stack by an encapsulation function

The procedures defined in this Recommendation permit dynamic negotiation of the encapsulation method and in case of mono-protocol encapsulation, the dynamic negotiation of the encapsulated protocol.

In the example of Figure 4:

– encapsulation function A knows that it needs to encapsulate the protocol Pk and is able to terminate all the protocols from protocol Pl to the physical layer;

– encapsulation function B knows that it needs to encapsulate the protocol Pl and is able to terminate all the protocols from protocol Pm to the physical layer.



NOTE – These protocols are terminated by encapsulation functions.

FIGURE 4/X.37

Dynamic negotiation of the encapsulated protocol

As the calling side, encapsulation function A requests the encapsulation of protocol Pk. As the called side, encapsulation function B responds with the encapsulation of protocol Pl since it does not know about protocol Pk but needs to encapsulate protocol Pl. According to X.37 negotiation procedures, protocol Pl will be encapsulated over the X.25 virtual call.

Encapsulation function A is able to terminate protocol Pm. If it is also able to encapsulate protocol Pl within X.25, encapsulation function A then accepts the negotiation result.

Since the initial need was to encapsulate protocol Pk, encapsulation function A may discard protocol Pl PDUs received from the local interface which carry PDUs of protocols different from protocol Pk in order to limit the amount of data sent on the X.25 virtual call. Since encapsulation function B has no knowledge of protocols used in the layers higher than the protocol Pl one, it encapsulates protocol Pl PDUs without additional filter. Upon reception of protocols Pl PDUs from the X.25 virtual call, encapsulation function A may again discard PDUs which carry PDUs of protocols different from protocol Pk.

The capability to negotiate the encapsulated protocol does not guarantee that the use of the established X.25 virtual call will be satisfactory and will answer to the identified need. For instance if the protocol stacks between protocols Pk and Pl are different on the two involved local interfaces, PDU transfer will fail.

6 Call Set-up procedures

This clause defines procedures used to set up an X.25 virtual call between two encapsulation functions independently of the protocol being encapsulated.

6.1 General

The use of the fast select acceptance facility is required by the encapsulation function.

The support of the encapsulation method and protocol negotiation is mandatory.

The support of either:

- mono-protocol encapsulation; or
- multi-protocol encapsulation; or
- mono-protocol and multi-protocol encapsulations,

is mandatory.

The support of the transfer method negotiation is mandatory.

The support of either:

- mono-PDU transfer method; or
- multi-PDU transfer method; or
- mono-PDU transfer and multi-PDU transfer methods,

is mandatory.

In case of multi-PDU transfer method, the support of a minimal delimiter length equal to two octets is mandatory.

6.2 Call set-up conditions

The criteria to set-up a virtual call are encapsulation function dependent. In particular any combination of the following criteria may be used:

- Encapsulation function power on;
- Availability of the local interface based on the physical level and or upper layers;
- Reception of PDUs from the local interface to be encapsulated.

6.3 Coding of the call request packet

6.3.1 Called DTE address

The called DTE address field and possibly, the called address extension facility are used to identify at least the called encapsulation function and the called port of the called encapsulation function (see the case of Frame Relay in clause 9).

When the encapsulation function is outside the network, several network dependent capabilities may be used, for instance:

- use of complementary addresses as defined in Appendix IV/X.25;
- allocation by the network of several X.121 addresses to the encapsulation function.

6.3.2 Calling DTE address

The calling DTE address field and/or possibly the contents of the calling address extension facility identify at least the calling port of the calling encapsulation function (see the case of Frame Relay in clause 9).

6.3.3 Facilities

The calling encapsulation function should request the fast select facility with no restriction on response, to enable the called encapsulation function to send called user data in the call accepted packet.

If the call is cleared with the cause fast select acceptance not subscribed, the encapsulation function may issue a second call without the fast select facility.

NOTE – Although as stated in 6.1, the use of the fast select acceptance facility is required by the encapsulation function, it is possible to set-up and use an X.25 virtual call with encapsulation function which does not operate according to this Recommendation on the fast select acceptance facility point. Negotiation procedures resolve the case where called user data is absent from the call connected packet.

For called and calling address extension facilities, see 6.3.1 and 6.3.2 above.

6.3.4 Call user data

The call user data field may contain the following fields:

- A protocol identifier which specifies the protocol encapsulation method and in case of mono-protocol encapsulation, the encapsulated protocol.
- A PDU transfer type which specifies the PDU transfer method and in case of multi-PDU transfer, the length of the delimiter.
- An additional field whose use may be defined in other Recommendations. Any use of the additional field according to this Recommendation is for further study.

The possible formats of the call user data are the following:

- one protocol identifier;
- one protocol identifier followed by one PDU transfer type;
- one protocol identifier followed by one PDU transfer type, followed by one additional field.

Consequently a calling encapsulation function shall at least encode the protocol identifier field in call user data.

6.3.4.1 Protocol identifier

The protocol identifiers are defined in ISO/IEC TR 9577 (second edition). All users of this Recommendation are encouraged to investigate the possibility of applying the most recent edition of the ISO/IEC TR 9577.

The protocol identifier field is of variable-length. Its length is implicitly determined by the first octet.

1) *Protocol encapsulation method*

   A non zero value of the protocol identifier means that the mono-protocol encapsulation is requested. A protocol identifier equal to zero means that the multi-protocol encapsulation is requested.

2) *Protocol identifier*

   In case of mono-protocol encapsulation, the protocol identifier specifies the protocol that will be carried over the X.25 virtual call.

   Note that when the code point 80 Hexadecimal [for IEEE SNAP (Subnetwork Access Protocols) convention] is used, the protocol identifier octet is extended to six: three additional octets for the Organisationally Unique Identifier (OUI) and two additional octets for the protocol identifier. These additional five octets are inserted in the call user data field after the protocol identifier octet and before the PDU transfer type octet, if present.

   A calling encapsulation function capable of both the mono-protocol encapsulation and the multi-protocol encapsulation method may encode a zero value in the protocol identifier in order to maximize the chances for a successful negotiation procedure if the calling encapsulation function has no particular requirement about either the protocol encapsulation method or the encapsulated protocol.

   A protocol identifier code point equal to A0 Hexadecimal is defined in the case the calling encapsulation function wants to specify that the protocol encapsulation method or the encapsulated protocol is determined by other means (e.g. administrative procedures). In particular this code point may be used by an encapsulation function which encapsulates HDLC based protocols and which terminates only the framing procedures: delimiting and possibly bit error detection with processing of the CRC. Namely, the encapsulation function encapsulates in X.25 data packets "what is between two flags" with or without CRC. The actual relevance of such encapsulation cannot simply rely on the protocol identifier, hence the use of the "identification through other means" protocol identifier code point.

6.3.4.2 PDU transfer type

The format of the PDU transfer type octet is given in Figure 5.

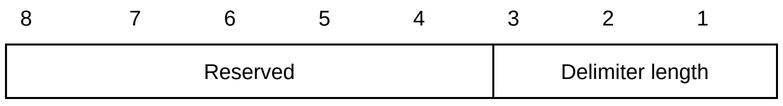


FIGURE 5/X.37

Format of the PDU transfer type octet

The delimiter length field contains the number of octets of the delimiter. A zero value in the delimiter length field means that there is no delimiter and consequently that the mono-PDU transfer is used. A non- zero value in the delimiter length field means that there is a delimiter and consequently that the multi-PDU Transfer is used. In this case, the delimiter length must be set to a minimum value of two octets.

The reserved field is set to 0 by the transmitting encapsulation function and is not interpreted by the receiving encapsulation function (for further extension).

A calling encapsulation function capable of using both the mono-PDU transfer and the multi-PDU transfer may encode the multi-PDU transfer in the PDU transfer type in order to maximize the chances for a successful negotiation procedure if the calling encapsulation function has no particular requirement about the PDU transfer method.

6.4 Reception of the incoming call packet

The called encapsulation function uses the called DTE address field, the called address extension facility, the calling DTE address field, the calling address extension facility, the call user data field with protocol identifier and PDU transfer type octets and the additional field, to determine if it accepts or rejects the incoming call. Examples of conditions that my also be used, are given in paragraph 6.2, e.g. availability of the local interface. If the called encapsulation function is not able to interpret the additional field, it shall ignore the latter.

6.5 Coding of the call accepted packet

6.5.1 Called DTE address

No additional specification. See 6.3.1.

6.5.2 Calling DTE address

No additional specification. See 6.3.2.

6.5.3 Facilities

No additional specification.

6.5.4 Called user data

The possible formats of the called user data are the following:

- nothing;
- one protocol identifier;
- one protocol identifier followed by one PDU transfer type;
- one protocol identifier followed by one PDU transfer type, followed by one additional field.

Consequently, the called encapsulation function may then insert in the called user data field, protocol identifier and PDU transfer type octets in order to specify the protocol encapsulation and PDU transfer methods as described in 6.3.4.1 and 6.3.4.2.

The negotiation rules defined in 6.7 may also have an influence on the coding of the protocol identifier and PDU transfer type octets by the called encapsulation function.

6.6 Reception of the call connected packet

The calling encapsulation function uses the called user data field and the negotiation rules defined in 6.7 to determine if it eventually accepts or rejects the call.

6.7 Negotiation at call set-up

6.7.1 Encapsulation method and protocol negotiation

According to the values of the protocol identifiers which are exchanged in the call and called user data, Table 1 gives the encapsulation method that will be used on the X.25 virtual call and in the case of mono-protocol encapsulation, the protocol that will be encapsulated.

The support of the encapsulation method and protocol negotiation is mandatory.

If the calling encapsulation function is not able to support the result of the encapsulation method and protocol negotiation, it should clear the call.

The support of either:

- mono-protocol encapsulation; or
- multi-protocol encapsulation; or
- mono-protocol and multi-protocol encapsulations,

is mandatory.

TABLE 1/X.37

Encapsulation method and protocol negotiation

Protocol identifier in call user data (Note 1)	Protocol identifier in called user data			
	No called user data ⇒ No protocol identifier (Note 2)	Protocol P1 (⇒ Mono-protocol)	Protocol P2 P2 not equal to P1 (⇒ Mono-protocol)	Multi-protocol
Protocol P1 (⇒ Mono-protocol)	Protocol P1 (⇒ Mono-protocol)	Protocol P1 (⇒ Mono-protocol)	Protocol P1 (⇒ Mono-protocol)	Not allowed
Multi-protocol	Multi-protocol	Protocol P1 (⇒ Mono-protocol)	Protocol P2 (⇒ Mono-protocol)	Multi-protocol

NOTES

1 Although, it is stated in 6.3.4 that a calling encapsulation function shall at least encode in call user data the protocol identifier field, it is recognized that the use of an empty call user data field leads to the mono-protocol encapsulation of ITU-T Rec. X.224 | ISO/IEC 8073.

2 The called user data field may be absent in case the called encapsulation function did not generate it or in case the second call is issued without the fast select facility (see 6.3.3).

6.7.2 Transfer method negotiation

According to the PDU transfer type values which are exchanged in the call and called user data fields, Table 2 gives the transfer method that will be used on the X.25 virtual call.

The support of the transfer method negotiation is mandatory.

If the calling encapsulation function is not able to support the result of the transfer method negotiation, it should clear the call.

The support of either:

– mono-PDU transfer method; or

– multi-PDU transfer method; or

– mono-PDU transfer and multi-PDU transfer methods,

is mandatory.

TABLE 2/X.37

Transfer method negotiation

PDU transfer type in called user data	PDU transfer type in call user data		
	No PDU transfer type (Note 1)	Mono-PDU transfer	Multi-PDU transfer
No PDU transfer type	Mono-PDU transfer	Not allowed (Note 2)	Not allowed (Note 2)
Mono-PDU transfer	Mono-PDU transfer	Mono-PDU transfer	Not allowed
Multi-PDU transfer	Mono-PDU transfer (Note 3)	Mono-PDU transfer	Multi-PDU transfer

NOTES

1 The PDU transfer type field may be absent in case the called encapsulation function did not generate it or in case the second call is issued without the fast select facility (see 6.3.3).

2 For backward compatibility reasons with a calling encapsulation function which does not support the transfer method negotiation procedure, a called encapsulation function shall not include in the called user data a PDU transfer type if the latter is absent in the received call user data.

3 For backward compatibility reasons with a called encapsulation function which does not support the transfer method negotiation procedure, the multi-PDU transfer is only used when both encapsulation functions specify this transfer method in call/called user data.

6.7.2.1 Delimiter length negotiation

In case of multi-PDU transfer, the called encapsulation function specifies in the PDU transfer type octets a delimiter length greater than one octet and less or equal to the delimiter length which was proposed by the calling encapsulation function.

Encapsulation functions should support a minimal delimiter length of two octets.

Note that moreover, the calling encapsulation function should support any delimiter length between:

– the length it specified in the call user data; and

– the minimal length of two octets.

6.8 Call set-up collision

For some reasons as charging, encapsulation of connection oriented protocols, security matters, it might be necessary to restrict the number of established X.25 virtual call to one.

In such a case when a call request has been transmitted and the corresponding call connected packet or clear indication packet has not yet been received and an incoming call packet is received for the same purpose, one of the calls has to be cleared. Then the encapsulation function compares the addresses of both encapsulation functions and, if its address is greater than the address of the remote encapsulation function, clears the call it has generated and of course confirms the incoming call.

The comparison is processed digit by digit from the left to the right according to the X.121 format (without prefix, but with DNIC and possibly escape code), followed by the complementary address.

7 Call clearing procedures

This clause defines the procedures used to clear a X.25 virtual call between two encapsulation functions independently of the protocol being encapsulated.

The call clearing conditions are implementation dependent and may include in addition to the call clearing conditions defined in recommendation X.25, the following ones pertained to the encapsulation function:

– non-availability of the local interface based on the physical level and or upper layers;

– incompatibility about the results of encapsulation and/or transfer methods negotiations;

– incompatibility about the identified protocol in case of mono-protocol encapsulation;

– X.25 virtual call detected as additional;

– error procedures;

– non-transfer of PDUs (reception and transmission) for a certain amount of time on the X.25 virtual call.

This last criterion may be used when the encapsulation function want to clear the X.25 virtual call because PDUs will not be transferred over this X.25 virtual call in the next lapse of time. One method to determine this, is to have a timer to detect inactivity period. On time-out, the X.25 virtual call is simply cleared.

When such timer is used, it is possible to determine its minimum value. In case of mono-protocol encapsulation, this minimum value depends on the default value defined for the encapsulated protocol (see the case of IP protocol in 10.3). In case of multi-protocol encapsulation, this minimum value depends on the maximum value among the default values defined for the encapsulated protocols.

For each one of the above call clearing conditions, the coding of the clearing cause and diagnostic fields is defined in Annex B.

8 Data Transfer procedure

This clause defines the data transfer procedures used on an X.25 virtual call between two encapsulation functions independently of the protocol being encapsulated.

8.1 Reception of PDU from the local interface

The encapsulation function first checks the validity of the PDUs. The following tests may apply:

– bit error detection;

– coding verification according to knowledge of the protocol stack used on the local interface;

– coding verification according to the encapsulated protocol specification;

– right to be sent on the X.25 virtual call based on lists of protocols in case of multi-protocol encapsulation and/or lists of addresses being used by the encapsulated protocol.

According to these tests, only valid PDUs are encapsulated over the X.25 virtual call.

8.2 Encapsulation format

Fields used by protocols of inferior layers on the local interface are not encapsulated.

Fields used for PDU delimitation (e.g. flags for HDLC) are not encapsulated.

Fields used for error bit detection are not encapsulated.

8.2.1 Mono-protocol encapsulation

No additional fields are added to the PDU before transfer on the X.25 virtual call.

8.2.2 Multi-protocol encapsulation

Before each PDU, a protocol identifier is added. Format, coding and meaning of this protocol identifier are identical to those defined in 6.3.4.1.

Figure 6 provides some examples of multi-protocol encapsulation.

Example 1: X.233 | ISO/IEC 8473-1, PDU directly identified through the protocol identifier

	Protocol identifier	PDU
Hexadecimal	1000 0001	ISO CLNP | X.233 PDU

Example 2: Identification of a proprietary protocol using the IEEE defined "SNAP" convention
[see Annex D / ISO/IEC TR 9577 (second edition)]

	Protocol identifier ↓	SNAP octets					PDU
Hexadecimal	80	00	00	00	81	37	IPX Datagram

FIGURE 6/X.37

Multi-protocol encapsulation examples

8.3 Data transfer on the X.25 virtual call on the network side

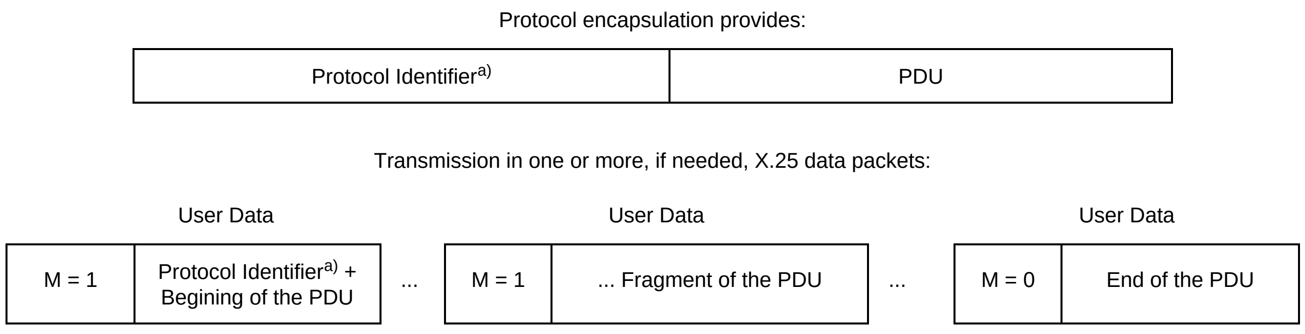
8.3.1 Mono-PDU transfer

Each PDU, plus its protocol identifier in case of multi-protocol encapsulation, is transferred in a complete packet sequence as shown in Figure 7.

8.3.2 Multi-PDU transfer

8.3.2.1 Principles

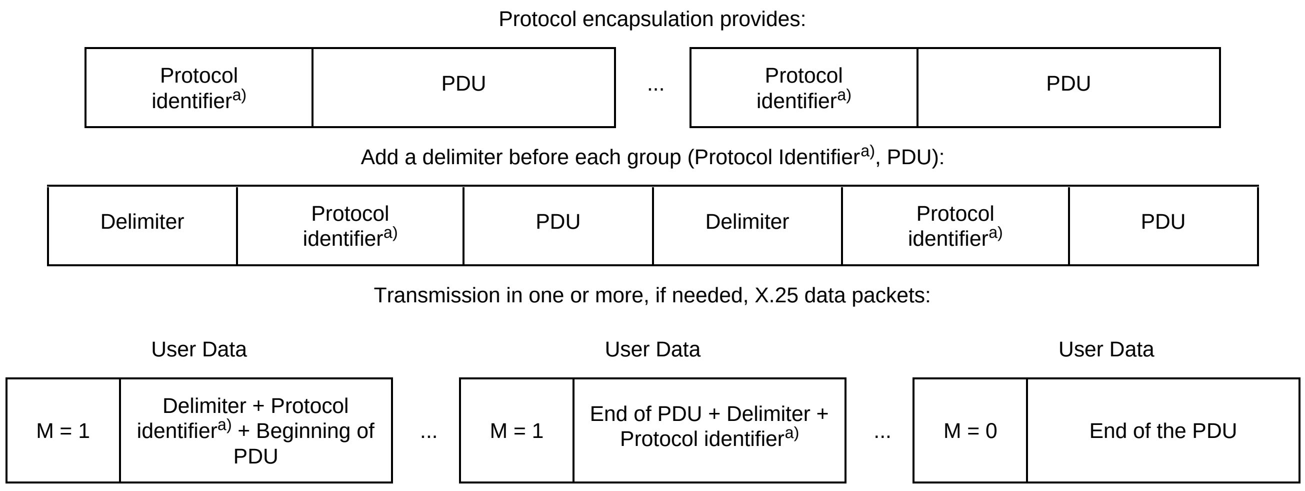
One or more PDUs are transferred in a complete packet sequence. Each PDU plus its protocol identifier in case of multi-protocol encapsulation, is preceded by a delimiter as showed in Figure 8.


Protocol encapsulation provides:
Protocol Identifier a)
PDU
Transmission in one or more, if needed, X.25 data packets:
User Data
User Data
User Data
M = 1
Protocol Identifier a) + Begining of the PDU
...
M = 1
... Fragment of the PDU
...
M = 0
End of the PDU

[a] This field is absent if the mono-protocol encapsulation method is used.

FIGURE 7/X.37

Example of Mono-PDU transfer


Protocol encapsulation provides:
Protocol identifier a)
PDU
...
Protocol identifier a)
PDU
Add a delimiter before each group (Protocol Identifier a), PDU):
Delimiter
Protocol identifier a)
PDU
Delimiter
Protocol identifier a)
PDU
Transmission in one or more, if needed, X.25 data packets:
User Data
User Data
User Data
M = 1
Delimiter + Protocol identifier a) + Beginning of PDU
...
M = 1
End of PDU + Delimiter + Protocol identifier a)
...
M = 0
End of the PDU

[a] This field is absent if the mono-protocol encapsulation method is used.

FIGURE 8/X.37

Example of Multi-PDU transfer

8.3.2.2 Delimiter format

Figure 9 defines the format of the delimiter:

– The reserved field may be used to carry some protocol information (see the frame relay example in clause 9). When not used, the reserved field is set to 0 by the encapsulation function and not interpreted (for further extension);

– The length field contains in octets the length of the group (Protocol identifier, PDU);

– The total length of the delimiter is determined at call set-up (see 6.7.2.1).

4 bits	[(8 × Delimiter length) – 4] bits
Reserved	Length

FIGURE 9/X.37

Delimiter format

8.3.2.3 Mode of operation

The criterias for an encapsulation function to assemble several PDUs in the same complete packet sequence are implementation dependent. A discussion on this subject is given in Appendix II.

8.4 Flow control

This subclause describes actions which may be undertaken by an encapsulation function when there is flow control either from the network side or from the local interface side.

8.4.1 Flow control on the X.25 virtual call from the network side

Upon flow control on an X.25 virtual call from the network side, an encapsulation function may, depending on the implementation:

- In case of multi-PDU transfer, assemble more than one PDU in a complete packet sequence.
- Send on the local interface some particular messages or signals according to the local protocol stack used, to prolong the X.25 flow control. If these messages or signals have a global meaning for the local interface, it may not be satisfactory, since the encapsulation function may observe a strong flow control on a given X.25 virtual call, and normal operation on another.
- Discard PDUs received from the local interface and candidate for transmission on the X.25 virtual call.

A discussion on this subject is given in Appendix II.

8.4.2 Flow control from the local interface side

Upon flow control from the local interface, an encapsulation function may, depending on the implementation:

- reduce the number of PDUs it sends on the local interface possibly on global basis or if the encapsulation function has sufficient knowledge, reduce the number of PDUs which are really affected by the flow control;
- impact this flow control on the X.25 side by reducing and even closing the windows of X.25 virtual calls.

9 Encapsulation of Frame Relay

This clause defines the X.25 call set-up procedures and the data transfer phase which follows, when an encapsulation function encapsulates Frame Relay.

9.1 General Principles

The general procedures described in clauses 6, 7 and 8 are applicable taking into account the following assumptions:

- the Frame Relay interface is described in Recommendation X.36;
- the mono-protocol encapsulation and the multi-PDU transfer are used.

NOTE – The case of the network-to-network interface as described in Recommendation X.76 can be easily derived from the X.36 case since the support of bi-directional management procedures is described.

9.1.1 Mapping of the circuits

For data transfer procedures, there is a mapping of one frame relay virtual circuit to one X.25 virtual call.

On the frame relay interface (see Recommendation X.36), the frame relay virtual circuit is identified by the DLCI.

On the X.25 interface, the virtual call is identified by the logical channel number.

When the encapsulation function is inside the network, the X.25 interface may be replaced by equivalent internal functions.

NOTE – The mapping of more than one frame relay virtual circuit to one X.25 virtual call is for further study.

9.1.2 Throughput class of the X.25 virtual call

The throughput class of the X.25 virtual call is inferred from the CIR (see Recommendation X.36) of the frame relay virtual circuit. When the CIR is set to 0, the value of the throughput class is implementation dependent.

9.2 Procedures for Frame Relay permanent virtual circuit

9.2.1 Call set-up procedures

9.2.1.1 Call set-up conditions

The criteria to set-up a virtual call are encapsulation function dependent. In particular the following criteria may be used:

- encapsulation function power on;
- availability of the Frame Relay interface based on the physical level;
- availability of the Frame Relay Interface based on link integrity verification procedures defined in clause 11/X.36;
- availability of the Frame Relay permanent virtual circuit, according to a status message received from the local interface when the encapsulation function acts as an X.36 DTE or when bi-directional management procedures are supported on the FR interface as defined in 11.5/X.36;
- reception of frames from the Frame Relay interface;
- etc.

9.2.1.2 Coding of the call request packet

9.2.1.2.1 Called DTE address

The called DTE address and possibly a called DTE address extension facility is used to identify:

- the called encapsulation function and the called port (FR interface) of the called encapsulation function;
- a DLCI number.

Figure 10 describes a case where the encapsulation functions are outside the network. To support the FR permanent virtual circuit identified on port i of encapsulation function (1), by DLCI n, and on port j of encapsulation function (2), by DLCI k, both encapsulation functions have to maintain a called DTE address which uniquely identifies the remote encapsulation function, port and DLCI.

If the main address and complementary addresses are used, the main address may identify the called encapsulation function and the complementary address may identify port and DLCI numbers.

9.2.1.2.2 Calling DTE address

The calling DTE address field and/or possibly the contents of the calling address extension facility identifies at least the calling port (FR interface) of the calling encapsulation function and a calling DLCI number.

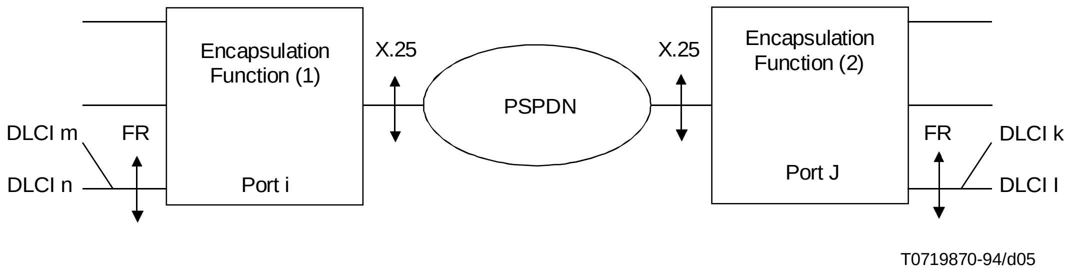


FIGURE 10/X.37

Use of called DTE address to identify an encapsulation function, a port and a FR permanent virtual circuit

9.2.1.2.3 Facilities

The calling encapsulation function should request the fast select facility with no restriction on response.

For called and calling address extension facility see 9.2.1.2.1 and 9.2.1.2.2 above.

9.2.1.2.4 Call user data

The call user data contains: the protocol identifier octet, the PDU transfer type octet and possibly the additional field.

The value of the protocol identifier octet for Frame Relay is hexadecimal.

The PDU transfer type shall encode a non-zero value, meaning that the multi-PDU transfer type is requested with a minimal delimiter length of two octets.

9.2.1.3 Reception of an incoming call packet

The called encapsulation function uses the called DTE address, the called address extension facility, the calling DTE address, the calling address extension facility, call user data with protocol identifier and PDU transfer type octets or not, to determine if it can accept or reject the incoming call. In particular, if the fast select facility is missing, the call has to be cleared. Some of the conditions defined in 10.2.1 may also be used.

9.2.1.4 Coding of the call accepted packet

No additional specification to the general procedures are needed for the called DTE address, the calling DTE address and facilities. The called user data field is encoded as follows:

– the protocol identifier octet is equal to hexadecimal A8;

– the PDU transfer type octet encodes multi-PDU transfer and a minimal value of two octets for the delimiter length;

– and possibly, the additional field.

9.2.1.5 Reception of the call connected packet

No additional specification to the general procedures.

9.2.1.6 Negotiation at called set-up

The encapsulation method negotiation should lead to the mono-protocol encapsulation and frame relay as the identified protocol.

The transfer method negotiation should lead to the multi-PDU transfer method.

9.2.1.7 Call set-up collision

As Frame Relay is a connection oriented protocol and as frame order delivery is guaranteed, it is necessary to restrict to one the number of established X.25 virtual call for a given Frame Relay permanent virtual circuit. So the procedures described in 6.8 are applicable.

9.2.2 Call clearing procedures

The call clearing conditions are implementation dependent and may include the following:

- non-availability of the FR interface based on the physical level;
- non-availability of the FR interface based on link integrity verification procedures defined in clause 11/X.36;
- non-availability of the FR permanent virtual circuit, according to a status message received from the local interface when the encapsulation function acts as an X.36 DTE or when bi-directional management procedures are supported on the FR interface as defined in 11.5/X.36;
- negotiations at call set-up do not lead to the mono-protocol encapsulation with FR as identified protocol;
- negotiations at call set-up do not lead to the multi-PDU transfer with a minimal delimiter length of two octets;
- X.25 virtual call detected as additional;
- non-transfer of PDUs (reception and transmission) for a certain amount of time on the X.25 virtual call;
- etc.

For each call clearing conditions, the coding of the clearing cause and diagnostic fields is defined in Annex B.

9.2.3 Support of management procedures

9.2.3.1 Support of management procedures when acting as an X.36 DCE

When the encapsulation function acts as an X.36 DCE, the support of Frame Relay permanent virtual circuit management procedures as described in clause 11/X.36, has the following impact on call set-up and call clearing procedures:

- the availability of the Frame Relay Interface based on link integrity verification procedures triggers the set-up of the X.25 virtual calls corresponding to the FR permanent virtual circuits supported on the Frame Relay Interface;
- the non-availability of the FR interface based on link integrity verification procedures triggers the clearing of the X.25 virtual calls corresponding to the FR permanent virtual circuits supported on the Frame Relay Interface.

Status of FR permanent virtual circuits

In the initial state, the encapsulation function sets the New bit to one and the Active bit to zero in all the PVC status information elements.

For a given FR permanent virtual circuit, when its X.25 virtual call is established, the encapsulation function sets the Active bit to one in the corresponding PVC status information element.

For a given FR permanent virtual circuit, when its X.25 virtual call is cleared with cause/diagnostic: DTE originated/Inactivity on the X.25 virtual call Time-out, the encapsulation function sets the Active bit to one in the corresponding PVC status information element. For all other clearing cause and diagnostic, the encapsulation function sets the Active bit to zero in the corresponding PVC status information element.

9.2.3.2 Support of management procedures when acting as an X.36 DTE

When the encapsulation function acts as an X.36 DTE, the support of Frame Relay permanent virtual circuit management procedures as described in clause 11/X.36, has the following impact on call set-up and call clearing procedures:

- The availability of the Frame Relay Interface based on link integrity verification procedures and the Active bit set to one in a PVC status information element received by the encapsulation function triggers the call set-up of the X.25 virtual call corresponding to the FR permanent virtual circuit supported on the Frame Relay Interface.
- The non-availability of the FR interface based on link integrity verification procedures triggers the clearing of the X.25 virtual calls corresponding to the FR permanent virtual circuits supported on the Frame Relay Interface.
- The Active bit set to zero in a PVC status information element received by the encapsulation function triggers the call clearing of the corresponding X.25 virtual call with cause/diagnostic: DTE originated/clear signalling received from the local interface.

9.2.3.3 Support of bi-directional management procedures

The support of bi-directional procedures for Frame Relay permanent virtual circuit management as described in 11.5/X.36, has the following impacts on call set-up and call clearing procedures:

- The availability of the Frame Relay Interface based on link integrity verification procedures and the Active bit set to one in a PVC status information element received by the encapsulation function triggers the call set-up of the X.25 virtual call corresponding to the FR permanent virtual circuit supported on the Frame Relay Interface.
- The non-availability of the FR interface based on link integrity verification procedures triggers the clearing of the X.25 virtual calls corresponding to the FR permanent virtual circuits supported on the Frame Relay Interface.
- The Active bit set to zero in a PVC status information element received by the encapsulation function triggers the call clearing of the corresponding X.25 virtual call with cause/diagnostic: DTE originated/clear signalling received from the local interface.

Status of FR permanent virtual circuits

In addition to the appropriate specifications given in 9.2.3.1, the following applies. For a given FR permanent virtual circuit, when the encapsulation function clears the corresponding X.25 virtual call with cause/diagnostic: DTE originated/clear signalling received from the local interface, the encapsulation function set the Active bit to one in the corresponding PVC status information element sent on the Frame Relay interface as described in Figure 11.

9.3 Procedures for switched virtual call

For further study.

9.3.1 Call set-up procedures

For further study.

9.3.2 Call clearing procedures

For further study.

9.4 FR frames transfer procedure

This subclause is applicable to both FR switched virtual circuit and permanent virtual circuit cases.

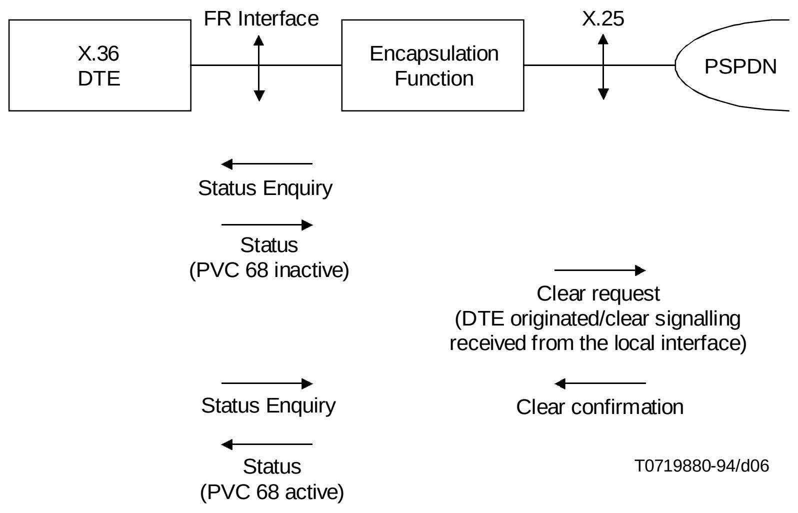


FIGURE 11/X.37

Status of a FR PVC upon clear signalling received from the FR interface

9.4.1 FR frames reception from the FR interface

The encapsulation function first checks the validity of the FR frames. For this purpose, the following tests are used:

- verification of the FR frames as defined in 9.4.4/X.36;
- right to be sent on the X.25 virtual call based on list of DLCI for whom an X.25 virtual call is configured/set-up.

According to these tests, only valid FR frames are encapsulated over X.25 virtual calls.

9.4.2 Encapsulation format

Flags, DLCI and error detection fields are not encapsulated.

9.4.3 FR frames transfer on the X.25 virtual call

The multi-PDU transfer is used along with a minimal delimiter length of two octets. FR header bits: FECN, BECN, DE, C/R, are transmitted in the delimiter field (see Figure 12).

Figure 13 defines the format of the delimiter for FR encapsulation:

- the length field contains in octets the length of the FR frame information field;
- the total length of the delimiter is determined at call set-up with a minimal value of two octets.

9.4.4 Flow control

This sublcause describes actions which may be undertaken in an encapsulation function when there is flow control either from the X.25 side or from the FR interface side.

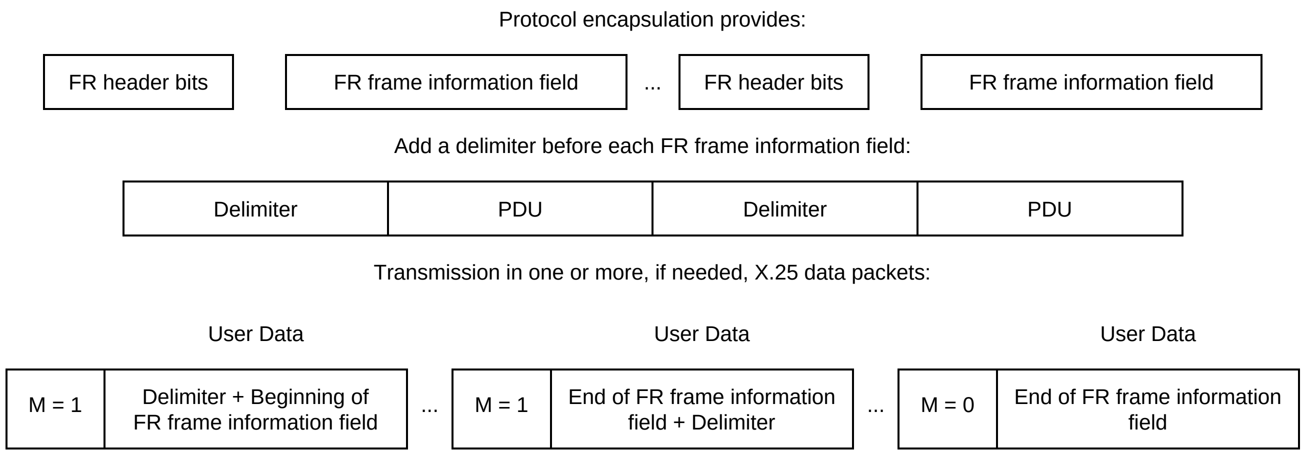

Protocol encapsulation provides:
FR header bits
FR frame information field
...
FR header bits
FR frame information field
Add a delimiter before each FR frame information field:
Delimiter
PDU
Delimiter
PDU
Transmission in one or more, if needed, X.25 data packets:
User Data
User Data
User Data
M = 1
Delimiter + Beginning of FR frame information field
...
M = 1
End of FR frame information field + Delimiter
...
M = 0
End of FR frame information field

FIGURE 12/X.37

Example of FR frames transfer

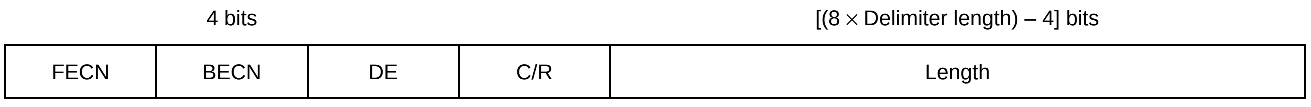


FIGURE 13/X.37

Delimiter format for Frame Relay encapsulation

9.4.4.1 Flow control from the X.25 virtual call

Upon flow control on an X.25 virtual call from the network side, an encapsulation function may, depending on the implementation:

- assemble more than one FR frame in a complete packet sequence, since the multi-PDU transfer is specified;
- in the FR frames of the corresponding FR virtual circuit and which are encapsulated on the X.25 virtual call, set to one the FECN bit;
- in the FR frame of the corresponding FR virtual circuit and which are sent on the local interface, set to one the BECN bit;
- discard FR frames of the corresponding FR virtual circuit, which are candidates for encapsulation on the X.25 virtual call and which have DE bit set to one;
- discard FR frames of the corresponding FR virtual circuit, which are candidates for encapsulation on the X.25 virtual call, regardless of the DE bit value.

A discussion on this subject is given in Appendix II.

9.4.4.2 Flow control from the Frame Relay interface side

The general specifications (see 8.4.2), are applicable.

9.4.5 Congestion notification

An encapsulation function may set to one the FECN/BECN bits to notify to the remote or local X.36 DTE congestion situation. Congestion situation may occur upon flow control from the X.25 virtual call (see 9.4.4.1) or because of lack of resources (e.g. memory, etc.) which are global to an encapsulation function, or for any other implementation dependent reasons.

10 Encapsulation of IP datagrams

This clause defines the X.25 call set-up procedures and the data transfer phase which follows, when an encapsulation function encapsulates IP datagrams.

10.1 General Principles

The general procedures described in clauses 6, 7 and 8 are applicable taking into account the following assumptions:

- The Internet Protocol Specification is described in RFC 791.
- X.25 virtual calls are set-up on demand. This means that the reception of IP datagrams from the local interface to be encapsulated, is used as a criteria to trigger the call set-up of the X.25 virtual call and that the non-transfer of IP datagrams for a certain amount of time, called period of inactivity triggers the call clearing of the X.25 virtual call.
- An X.25 virtual call is cleared after some period of inactivity.
- Both mono-protocol and multi-protocol encapsulation methods may be used.
- Both mono-PDU and multi-PDU transfer methods may be used.

10.2 Call set-up procedures

10.2.1 Call set-up conditions

Reception of IP datagrams to be encapsulated shall be used as a criterion to trigger the call set-up of X.25 virtual calls.

10.2.2 Coding of the call request packet

No additional specification to the general procedures are needed for the called DTE address, the calling DTE address and the facilities.

The call user data field is encoded as follows:

- In case of mono-protocol encapsulation, the protocol identifier is equal to CC hexadecimal [IP may also be theoretically identified through the IEEE SNAP convention (OUI: 00-00-00 hexadecimal, PID: 80-00 hexadecimal)], however this use is not advisable for the mono-protocol encapsulation within this Recommendation.
- In case of multi-protocol encapsulation, the protocol identifier is equal to 00 hexadecimal as specified in the general procedures.
- For the PDU transfer type, both mono-PDU and multi–PDU transfer methods may be requested.
- An additional field may be inserted.

10.2.3 Reception of an incoming call packet

No additional specification to the general procedures are needed.

Although, in case of mono-protocol encapsulation the use of the IEEE SNAP convention to identify IP is not advisable, encapsulation functions may interpret this identification mechanism.

10.2.4 Coding of the call accepted packet

No additional specification to the general procedures are needed for the called DTE address, the calling DTE address and facilities.

If encoded, the called user data field should be encoded like the call user data field in the call request packet (see 10.2.2).

10.2.5 Reception of the call connected packet

No additional specification to the general procedures.

Although, in case of mono-protocol encapsulation the use of the IEEE SNAP convention to identify IP is not advisable, encapsulation functions may interpret this identification mechanism.

10.2.6 Negotiation at called set-up

No additional specification to the general procedures.

10.2.7 Call set-up collision

More than one X.25 virtual call may be established between two encapsulation functions for the same IP encapsulation purpose.

However, an encapsulation function may require to restrict the number of established X.25 virtual call to one. In such a case, the procedures described in 6.8 are applicable.

10.3 Call clearing procedures

The detection of some period of inactivity on an X.25 virtual call shall be used as a criteria to trigger the call clearing of that particular X.25 virtual call.

The inactivity timer should be configurable. In case of encapsulation of IP datagrams, the default value for this timer is 90 seconds.

See Annex B for the coding of the clearing cause and diagnostic fields.

10.4 IP datagrams transfer procedure

Encapsulation function must be able to receive and transmit IP datagrams up to at least 1600 octets in length. This enables the standard IP Maximum Transmission Unit (MTU) of 1500 octets to be used both on local interface and on X.25 virtual call. If so, no fragmentation using IP procedures is necessary at the encapsulation function.

NOTE – The IP maximum Transmission Unit (MTU) is totally independent of the packet length used by the encapsulation function on the X.25 interface since complete packet sequences are used to transmit IP datagrams.

For backward compatibility, the transmit Maximum Transmission Unit for IP datagram should default to 1500 octets and should be configurable at least in the range 576 to 1600 octets.

Encapsulation function may support maximum IP datagram size larger than 1600 octets.

10.4.1 IP datagrams reception from the local interface

The encapsulation function first checks the validity of the IP datagrams. For this purpose, the following tests are used:

- verification of the IP datagrams as defined in RFC 791;
- right to be sent on the X.25 virtual call which may be based on the examination of source and destination addresses of the IP datagrams.

According to these tests, only valid IP datagrams are encapsulated over X.25 virtual calls.

10.4.2 Encapsulation format

The complete IP datagram is encapsulated.

Figure 14 gives the different encapsulation format which may be used according to the encapsulation method negotiated.

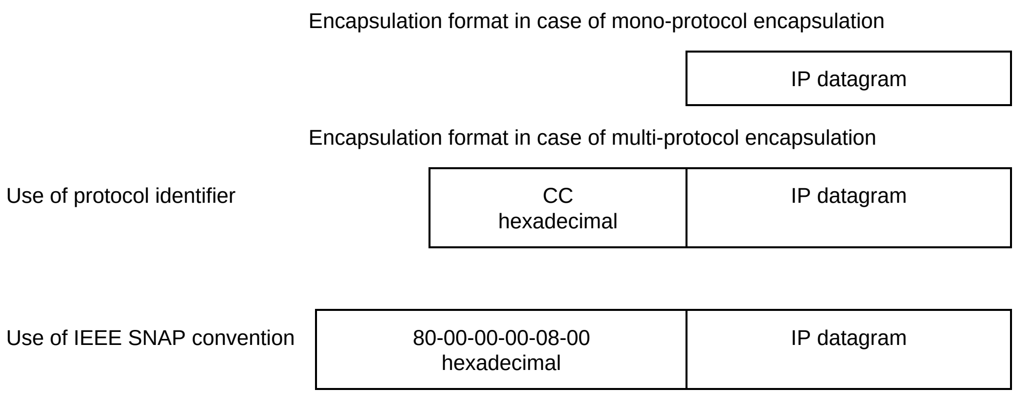

Encapsulation format in case of mono-protocol encapsulation
IP datagram
Encapsulation format in case of multi-protocol encapsulation
Use of protocol identifier
CC hexadecimal
IP datagram
Use of IEEE SNAP convention
80-00-00-00-08-00 hexadecimal
IP datagram

FIGURE 14/X.37

IP encapsulation formats

10.4.3 IP datagrams transfer on the X.25 virtual call

Subclause 8.3 is applicable taking into account that:

- both mono-PDU and multi-PDU transfer methods may be used;
- in case of multi-PDU transfer, the reserved field of the delimiter is not used and consequently set to 0 by the sending encapsulation function and not interpreted by the receiving encapsulation function.

10.5 Flow control

The general procedures described in 8.4 are applicable.

Annex A

Protocol Implementation Conformance Statement (PICS) proforma[2)]

(This annex forms an integral part of this Recommendation)

A.1 Instructions for completing the PICS proforma

A.1.1 Description of the PICS proforma

In order to reduce the size of the tables in the PICS proforma, notations have been introduced. These have allowed the use of multi-column layout where the columns are Item No., Protocol Feature, Status, Support Reference to X.37 and Comments. The definition of each of these is given in the next subclauses.

This PICS Proforma has been divided into the following main clauses:

a) Identification;

b) Claim conformance to standards;

c) Call set-up procedures;

d) Call clearing procedures;

e) Data transfer procedures;

f) Frame relay encapsulation;

g) IP datagrams encapsulation.

A.1.2 Item Number Column

This column contains a serial number that increases monotically down the table to enable reference to a row of the table.

A.1.3 Protocol Feature

This column lists the Protocol Features of Recommendation X.37. The supplier or the implementor is required to fill in the PICS in respect of these entries.

A.1.4 Reference column (Ref)

This column gives references to clauses in Recommendation X.37.

A.1.5 Status column (stat)

This column indicates the level of support required for conformance to Recommendation X.37. These are detailed below:

'm'	Mandatory support is required for conformance to Recommendation X.37
'o'	Optional support is required for conformance to Recommendation X.37. If implemented, it must conform to the specifications contained in Recommendation X.37
'o.n'	The notation o.<n> signifies that at least one out of the n group shall be implemented (where <n> is a positive integer)
'cn'	Conditional support as indicated by the predicate expression for cn (where <n> is a positive integer)
'x'	Use is prohibited
'n/a'	Indicates that the item is not applicable
'-'	Out of scope

A.1.6 Support column (Supp)

The "Support" column shall be completed by the supplier or implementor to indicate the level of implementation of each feature. Where a column is pre-printed with "n/a"; representing a non-applicable entry, no entry shall be inserted at that position. When in a Status column the out of scope sign "-" is present, the Support column must be filled with "N".

[2)] Users of this Recommendation may freely reproduce the PICS proforma in this annex so that it can be used for its intended purpose, and may further publish the completed PICS.

Elsewhere PICS proforma has been designed such that the only entries required are:

"Y"	Yes the feature has been implemented. If "Y" is entered in a PICS table, the value of that entry when referenced in Boolean expression is "TRUE".
"N"	No, the feature has not been implemented. If "N" is entered in a PICS table, the value of that entry when referenced in Boolean expression is "FALSE".
"Ig"	Ignore, the item is not treated as protocol error, but is ignored rather than processed. If "Ig" is entered in a PICS table, the value of that entry when referenced in Boolean expressions is "FALSE".
"Err"	Error, the occurrence of this item is treated as a protocol error. It "Err" is entered in a PICS table, the value of that entry when referenced in a Boolean expression is "FALSE".

"Ig" and "Err" have the same static conformance semantic as "N". If an item is marked as "m" in the Status column then only "Y" may be checked in the Support column for the implementation to be conformant.

In the PICS proforma tables, every leading feature marked 'm' shall be supported by the implementation.

A.1.7 Comment column

This column may be pre-printed with an explanatory comment, or left blank for the implementor to add a comment on the response given, or other relevant information.

A.1.8 Unique identification of a clause

Each line, within a clause of PICS proforma, which requires implementation details to be entered, is numbered in the left hand box of the line. This numbering is included as a means of uniquely identifying all possible implementation detail within the PICS proforma. This referencing is used both inside the PICS proforma and for references from other Test Specification documents.

All response shall be referenced by specifying the following sequence:

a) a reference to the smallest subclause enclosing the relevant item;

b) a solidus character "/ ";

c) the reference number of the row in which the response appears.

A.1.9 Predicates

A predicate is an explicit reference to a PICS proforma YES/NO entry, or a relational expression involving a reference to a PICS proforma entry which gives a value as an answer, or a predicate expression (i.e. Boolean expression involving predicates).

A predicate uses clauses format as defined in the above A.3.8 and will interpret implementor's support answer as described in A.3.5.

Predicates are used in condition.

An example of a predicate is *NOT(A.4.1/4 AND A.4/3)*

A.1.10 Completion of the PICS proforma

The implementor shall complete all entries in the columns "Support" following the instruction of A.1.4.2. In addition, other specifically identified information shall be provided by the implementor where requested. No changes shall be made to the proforma except the completion as required, recognising that the level of detail required may, in some instances, exceed the space available for responses, additional responses may be given by the addition of appendices to the PICS.

All entries within the PICS proforma shall be made in ink. Alteration to such entries shall be made by crossing out, but not erasing nor making the original entry illegible, and writing the entry alongside. All such alterations to records shall be initiated by the staff making them.

A.2 Identification

A.2.0 General

This subclause is used to record the date of the completion of the PICS, and to describe the supplier of the implementation, the implementation itself and the Recommendation to which the implementation is claimed to conform.

A.2.1 Identification of the PICS

This subclause allows to univocally identify this document.

Item No.	Question	Response
1	Date of the statement (yy-mm-dd)	
2	PICS Serial Number	

A.2.2 Supplier and Implementation Details

This subclause allows for the specification of the information necessary to uniquely identify the implementation and the systems in which it may reside.

A.2.2.1 Supplier Details

This subclause allows for the identification of the supplier of the PICS. At least items 1, 2 and 3 shall be filled in.

Item No.	Question	Response
1	Organisation Name	
2	Contact Name(s)	
3	Address	
4	Telephone	
5	Telex Teletex	
6	Fax	
7	E-mail	
8	Other information	

A.2.2.2 Implementation Details

Specify the information necessary to uniquely identify the implementation and the systems in which it may reside. At least items 1 through 6 shall be filled in.

Item No.	Question	Response
1	Implementation Name	
2	Version	
3	Hardware Name	
4	Hardware Version	
5	Operating System Name	
6	Operating System Version	
7	Other Information	

A.3 Claimed Conformance to standards

A.3.1 Identification of the protocol

Item No.	Question	Response
1	Title, Reference Number and Date of publication of the Protocol Standard	

A.3.2 Global Conformance Statement

Item No.	Question	Response
1	Are all mandatory capability implemented?	

Answering “No” to this question indicates non-conformance to the protocol specification. Non-supported mandatory capabilities are to be identified in the PICS, with an explanation of why the implementation is non-conformant. Such information shall be provided in A.9, under “Other information”.

A.4 Call set-up procedures

Item No.	Protocol Feature	Reference	Stat	Supp	Comment
1	Encapsulation method and protocol negotiation	6.7.1	m		
2	Protocol encapsulation method	5.2.1			
3	Mono-Protocol encapsulation method	5.2.1.1	o.1		
4	Multi-Protocol encapsulation method	5.2.1.2	o.1		
5	Transfer method negotiation	6.7.2	m		
6	PDU transfer method	5.2.2			
7	Mono-PDU transfer method	5.2.2.1	o.2		
8	Multi-PDU transfer method	5.2.2.2	o.2		
9	Delimiter length negotiation	6.7.2.1	c1		
o.1 At least one among mono-protocol encapsulation, multi-protocol encapsulation, shall be supported o.2 At least one among mono-PDU transfer, multi-PDU transfer method, shall be supported c1 IF(A.4/8) THEN m ELSE n/a					

A.4.1 Calling side

Item No.	Protocol Feature	Reference	Stat	Supp	Comment
1	Request the fast select facility with no restriction on response	6.3.3	m		
2	Issue a second call without fast select facility when the first call is cleared with the cause fast select acceptance not subscribed	6.3.3	o		
3	Code protocol identifier in call user data	6.3.4	m		
4	Code PDU transfer type in call user data	6.3.4	o		
5	Code additional field in call user data	6.3.4	o		
6	Receive Protocol identifier in called user data	6.6	m		
7	Receive PDU transfer type in called user data	6.6	m		
8	Receive Additional field in called user data	6.6	m		

A.4.2 Called side

Item No.	Protocol Feature	Reference	Stat	Supp	Comment
1	Receive Protocol identifier in call user data	6.4	m		
2	Receive PDU transfer type in call user data	6.4	m		
3	Receive Additional field in call user data	6.4	m		
4	Code Protocol identifier in called user data	6.5.4	o		
5	Code PDU transfer type in called user data	6.5.4	o		
6	Code Additional field in called user data	6.5.4	o		

A.5 Call clearing procedures

Item No.	Protocol Feature	Reference	Stat	Supp	Comment
1	Restrict to one the number of X.25 virtual call established for a given purpose	6.8	o		
2	Address comparison to determine which X.25 virtual call will be cleared	6.8	c2		
3	Call clearing upon inactivity period detection	7	o		
4	Coding of clearing cause and diagnostic field	Annex B	m		
c2	IF(A.5/1) THEN m ELSE n/a				

A.6 Data transfer procedures

Item No.	Protocol Feature	Reference	Stat	Supp	Comment
1	Fragment PDU in a complete packet sequence when the mono-PDU transfer is used	8.3.1	o		
2	Receive fragmented PDU in a complete packet sequence when the mono-PDU transfer is used	8.3.1	m		
3	Gather several PDUs when the multi-PDU transfer is used	8.3.2	o		
4	Receive gathered PDUs when the multi-PDU transfer is used	8.3.2	m		
5	Fragment gathered PDUs in a complete packet sequence when the multi-PDU transfer is used	8.3.2	o		
6	Receive gathered PDUs which are fragmented in a complete packet sequence when the multi-PDU transfer is used	8.3.2	m		

A.6.1 Reaction to flow control from the X.25 virtual call

Item No.	Protocol Feature	Reference	Stat	Supp	Comment
1	Gather several PDUs when the multi-PDU transfer is used	8.4.1	o		
2	Send on the local interface particular messages or signals	8.4.1	o		
3	Discard PDUs received from the local interface	8.4.1	o		

A.6.2 Reaction to flow control from the local interface

Item No.	Protocol Feature	Reference	Stat	Supp	Comment
1	Reduce the number of PDUs sent on the local interface	8.4.2	o		
2	Impact the flow control from the local interface on the X.25 side	8.4.2	o		

A.7 Encapsulation of Frame Relay

Item No.	Protocol Feature	Reference	Stat	Supp	Comment
1	Encapsulation of FR frames in X.25 data packets	9	o		
2	Procedures for frame Relay permanent virtual circuit	9.2	o		

A.7.1 Call set-up procedures for Frame Relay

Item No.	Protocol Feature	Reference	Stat	Supp	Comment
1	Mono-protocol encapsulation method	9.1	c3		
2	Multi-PDU transfer	9.1	c3		
c3	IF(A.7/2) THEN m ELSE n/a				

A.7.1.1 Calling side

Item No.	Protocol Feature	Reference	Stat	Supp	Comment
1	Request the fast select facility with no restriction on response	9.2.1.2	c3		
2	Code protocol identifier to FR (hexadecimal A8) in call user data	9.2.1.2	c3		
3	Code PDU transfer type to Multi-PDU transfer in call user data	9.2.1.2	c3		
4	Code additional field in call user data	9.2.1.2	o		
c3 IF(A.7/2) THEN m ELSE n/a					

A.7.1.2 Called side

Item No.	Protocol Feature	Reference	Stat	Supp	Comment
1	Code protocol identifier to FR (hexadecimal A8) in called user data	9.2.1.4	c3		
2	Code PDU transfer type to Multi-PDU transfer in called user data	9.2.1.4	c3		
3	Code additional field in called user data	9.2.1.4	o		
c3 IF(A.7/2) THEN m ELSE n/a					

A.7.2 Call clearing procedures

Item No.	Protocol Feature	Reference	Stat	Supp	Comment
1	Restrict to one the number of X.25 virtual call established for a given purpose	9.2.1.7	c3		
2	Address comparison to determine which X.25 virtual call will be cleared	9.2.1.7	c4		
3	Call clearing upon inactivity period detection	9.2.2	o		
4	Coding of clearing cause and diagnostic field	Annex B	m		
c3 IF(A.7/2) THEN m ELSE n/a c4 IF(A.7.2/1) THEN m ELSE n/a					

A.7.3 Management procedures

Item No.	Protocol Feature	Reference	Stat	Supp	Comment
1	FR permanent virtual circuit management procedures	9.2.3	c3		
2	FR PVC management procedures as a DCE	9.2.3.1	o		
3	FR PVCmanagement procedures as a DTE	9.2.3.2	o		
4	FR permanent virtual circuit bi-directional management procedures	9.2.3.2	o		
5	Non-availability of the FR interface triggers call clearing	9.2.3	c5		
6	Availability of the FR interface triggers call set-up	9.2.3.1	c14		
7	Availability of the FR interface and Received Active bit set to one triggers call set-up	9.2.3.2 9.2.3.3	c6		
8	Received Active bit set to zero triggers call clearing	9.2.3.2 9.2.3.3	c6		
c3 IF(A.7/2) THEN m ELSE n/a c5 IF(A.7.3/1) THEN m ELSE n/a c14 IF(A.7.3/2) THEN m ELSE n/a c6 IF((A.7.3/3) OR (A.7.3/4)) THEN m ELSE n/a					

A.7.3.1 Status of FR permanent virtual circuit

Item No.	Protocol Feature	Reference	Stat	Supp	Comment
1	In the initial state, set the New bit to one, set the Active bit to zero	9.2.3.1	c15		
2	Set the Active bit to one when the corresponding X.25 virtual call is established	9.2.3.1	c15		
3	Set the Active bit to one when the corresponding X.25 virtual call is cleared with cause/diagnostic: DTE originated/Inactivity on the X.25 virtual call Time-out	9.2.3.1	c15		
4	Set the Active bit to zero when the corresponding X.25 virtual call is cleared with another cause/diagnostic than DTE originated/Inactivity on the X.25 virtual call Time-out	9.2.3.1	c15		
5	Set the Active bit to one when send the clear request sent upon received Active bit set to zero	9.2.3.3	c16		
c15 IF((A.7.3/2) OR (A.7.3/4)) THEN m ELSE n/a c16 IF(A.7.3/4) THEN m ELSE n/a					

A.7.4 FR frames transfer procedures

Item No.	Protocol Feature	Reference	Stat	Supp	Comment
3	Gather several FR frames	9.4.3	c7		
4	Receive gathered FR frames	9.4.3	c8		
5	Fragment gathered FR frames in a complete data packet	9.4.3	c7		
6	Receive gathered FR frames which are fragmented in a complete packet sequence	9.4.3	c8		
c7 IF(A.7/1) THEN o ELSE n/a c8 IF(A.7/1) THEN m ELSE n/a					

A.7.4.1 Reaction to flow control from the X.25 virtual call

Item No.	Protocol Feature	Reference	Stat	Supp	Comment
1	Gather several FR frames	9.4.4.1	c7		
2	Set the FECN bit to one in FR frames encapsulated on the X.25 virtual call	9.4.4.1	c7		
3	Set the BECN bit to one in FR frames sent on the FR interface	9.4.4.1	c7		
4	Discard FR frames to be encapsulated on the X.25 virtual call which have DE bit set to one	9.4.4.1	c7		
5	Discard FR frames to be encapsulated on the X.25 virtual call	9.4.4.1	c7		
c7 IF(A.7/1) THEN o ELSE n/a					

A.7.4.2 Reaction to congestion notification from the FR interface

Item No.	Protocol Feature	Reference	Stat	Supp	Comment
1	Reduce the number of PDUs sent on the FR interface	9.4.4.2	c7		
2	Impact the flow control from the FR interface on the X.25 side	9.4.4.2	c7		

A.7.4.3 Congestion notification

Item No.	Protocol Feature	Reference	Stat	Supp	Comment
1	Set the FECN bit to one upon congestion occurrence	9.4.5	c7		
2	Set the BECN bit to one upon congestion occurrence	9.4.5	c7		
c7 IF(A.7/1) THEN o ELSE n/a					

A.8 Encapsulation of IP datagrams

Item No.	Protocol Feature	Reference	Stat	Supp	Comment
1	Encapsulation of IP datagrams in X.25 data packets	10	o		

A.8.1 Call set-up procedures

Item No.	Protocol Feature	Reference	Stat	Supp	Comment
1	Protocol encapsulation method	10.1			
2	Mono-Protocol encapsulation method	10.1	o.1		
3	Multi-Protocol encapsulation method	10.1	o.1		
4	PDU transfer method	10.1			
5	Mono-PDU transfer method	10.1	o.2		
6	Multi-PDU transfer method	10.1	o.2		
o.1 At least one among mono-protocol encapsulation, multi-protocol encapsulation, shall be supported o.2 At least one among mono-PDU transfer, multi-PDU transfer method, shall be supported					

A.8.1.1 Calling side

Item No.	Protocol Feature	Reference	Stat	Supp	Comment
1	Reception of IP datagrams used as a criteria to trigger the call set-up of X.25 virtual call	10.2.1	c9		
2	Code protocol identifier in call user data to CC hexadecimal	10.2.2	c10		
3	Code protocol identifier in call user data to 00 hexadecimal	10.2.2	c11		
4	Code PDU transfer type in call user data	10.2.2	o		
5	Code additional field in call user data	10.2.2	o		
6	Receive IP Protocol identifier belonging to the IEEE SNAP convention in called user data	10.2.5	o		
c9 IF (A.8/1) THEN m ELSE n/a c10 IF (A.8.1/2) THEN m ELSE n/a c11 IF (A.8.1/3) THEN m ELSE n/a					

A.8.1.2 Called side

Item No.	Protocol Feature	Reference	Stat	Supp	Comment
1	Receive IP Protocol identifier belonging to the IEEE SNAP convention in call user data	10.2.3	o		

A.8.2 Call clearing procedures

Item No.	Protocol Feature	Reference	Stat	Supp	Comment
1	Restrict to one the number of X.25 virtual call established for the same IP encapsulation	10.2.7	o		
2	Address comparison to determine which X.25 virtual call will be cleared	10.2.7	c12		
3	Call clearing upon inactivity period detection	10.3	c9		
4	Configurable inactivity timer	10.3	c13		
c9 IF (A.8/1) THEN m ELSE n/a c12 IF(A.8.2/1) THEN m ELSE n/a c13 IF (A.8.2/3) THEN m ELSE n/a					

A.8.3 IP datagrams transfer procedures

Item No.	Protocol Feature	Reference	Stat	Supp	Comment
1	Default IP Maximum Transmission Unit (MTU) of 1500 octets	10.4	c9		
2	Configurable IP Maximum Transmission Unit (MTU) from 576 to 1600 octets	10.4	c9		
3	Higher IP Maximum Transmission Unit (MTU) than 1600 octets	10.4	o		
4	Verification of IP datagrams as defined in RFC 791	10.4.1	c9		
5	Source addresse examination	10.4.1	o		
6	Destination addresse examination	10.4.1	o		
7	Code/receive protocol identifier to CC hexadecimal or to IP Protocol identifier belonging to the IEEE SNAP convention	10.4.2	c11		
c9 IF (A.8/1) THEN m ELSE n/a c11 IF (A.8.1/3) THEN m ELSE n/a					

A.9 Other information

Item No.	Other Information

Annex B

Coding of clearing cause and diagnostic fields generated by an encapsulation function in clear request packet

(This annex forms an integral part of this recommendation)

For each clearing condition, this annex provides the coding of the clearing cause and diagnostic fields.

Table B.1 gives the coding of clearing cause and diagnostic fields for an encapsulation function part of the public network.

An encapsulation function not part of the public network must use the same code points for the clearing cause with bit 8 set to one, except when a value zero (“DTE originated”) is defined.

TABLE B.1/X.37

Coding of clearing cause and diagnostic field in clear request packet

Clearing condition	Cause	Diagnostics (Decimal)
Fault on the physical or logical level on the local interface	Out of order	00 or network specific
Unauthorised calling DTE address in the incoming call packet	Access bared	68
Unknown called DTE address in the incoming call packet	Not obtainable	67
X.25 virtual call detected as additional	DTE originated (value zero)	1
Mono-PDU transfer not accepted by the calling encapsulation function	DTE originated (value zero)	2
Error in call user data (delimiter length equal to 1; additional field in error or present while not expected)	DTE originated (value zero)	3
Violation of the negotiation rules	DTE originated (value zero)	4
Inactivity on the X.25 virtual call Time-out	DTE originated (value zero)	16
Clear signalling received from the local interface	DTE originated (value zero)	17
Delimiter too short or the length indicated in the delimiter is larger than the number of remaining octets in the complete packet sequence	DTE originated (value zero)	18
Attempt of intrusion detected on the encapsulated protocol level	DTE originated (value zero)	19
Reception of encapsulated PDU not conforming with the expected protocol	DTE originated (value zero)	20
Identified protocol unknown or not supported	DTE originated (value zero)	249

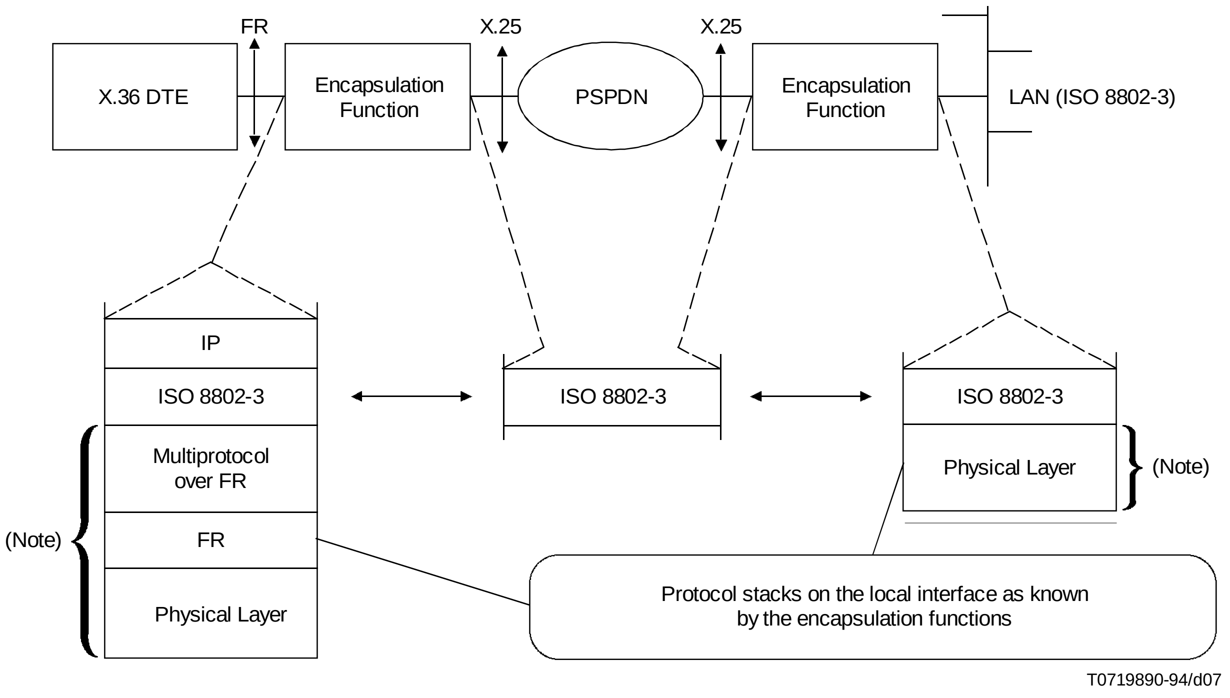
Appendix I

Application examples

(This appendix does not form an integral part of this Recommendation)

I.1 Dynamic negotiation of the encapsulated protocol

In the first example depicted in Figure I.1/X.37, the identified need is to transmit IP datagrams between an X.36 DTE and an ISO 8802-3 LAN.



NOTE – These protocols are terminated by encapsulation functions.

FIGURE I.1/X.37

Transmission of IP datagrams between a X.36 DTE and an ISO 8802-3 LAN

On the FR access, IP datagrams are encapsulated in ISO 8802-3 frames, which are in turn encapsulated in FR frames using the “multi-protocol over FR” mechanism (see clause 13/X.36).

On the ISO 8802-3 LAN, IP datagrams are encapsulated in ISO 8802-3 frames.

Encapsulation function (1) tries to set-up an X.25 call towards encapsulation function (2) with call user data specifying IP encapsulation and multi-PDU transfer.

Encapsulation function (2) is not able to support the IP encapsulation but can support the encapsulation of IS0 8802-3 frames. In such a case, encapsulation function (2) may answer with called user data specifying ISO 8802-3 frames encapsulation and multi-PDU transfer.

Encapsulation function (1) may then either clear the call, or begin data transfer procedures knowing that it encapsulates over the X.25 virtual call ISO 8802-3 frames which in turn contains IP datagrams.

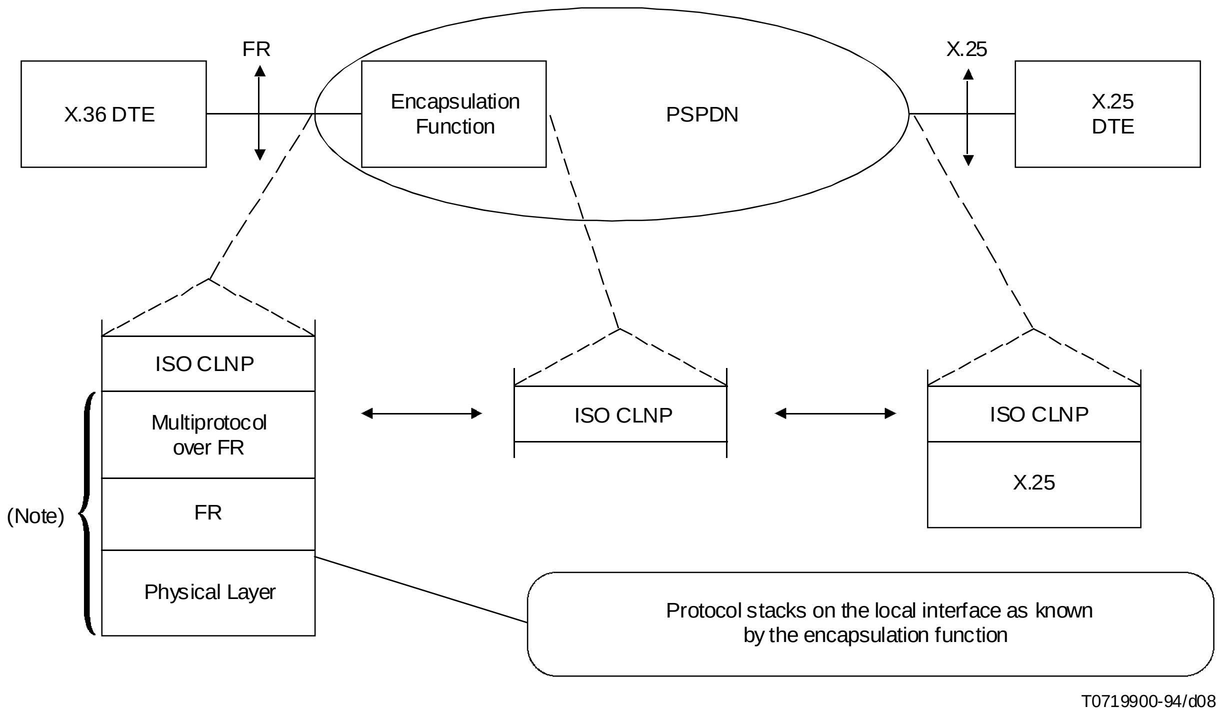
In this example, the ISO 8802-3 frames are carried in FR frames only between the X.36 DTE and the encapsulation function (1). This means that from the perspective of this FR virtual circuit, encapsulation function (1) is seen more as a DTE than an usual DCE. The following impacts on the procedures upon flow control from the X.25 virtual call described in 9.4.4.1, are:

– since no FR frames are transmitted on the X.25 virtual call, there is no FECN bit setting;

– since the FR frames are received by the final equipment of the FR connection, no interpretation of the DE bit is needed.

I.2 X.25 DTE / X.36 DTE interworking for a given encapsulated protocol

In the second example depicted in Figure I.2, the identified need is to transmit ISO CLNP datagrams between an X.36 DTE and X.25 DTE.



NOTE – These protocols are terminated by the encapsulation function.

FIGURE I.2/X.37

Transmission of ISO CLNP datagrams between a X.36 DTE and an X.25 DTE

The X.36 DTE encapsulates ISO CLNP PDUs using the "multi-protocol over FR" mechanism (see clause 13/X.36).

The X.25 DTE encapsulates ISO CLNP PDUs according to this Recommendation.

The encapsulation function tries to set-up an X.25 virtual call towards X.25 DTE with call user data specifying ISO CLNP, since it is the identified need.

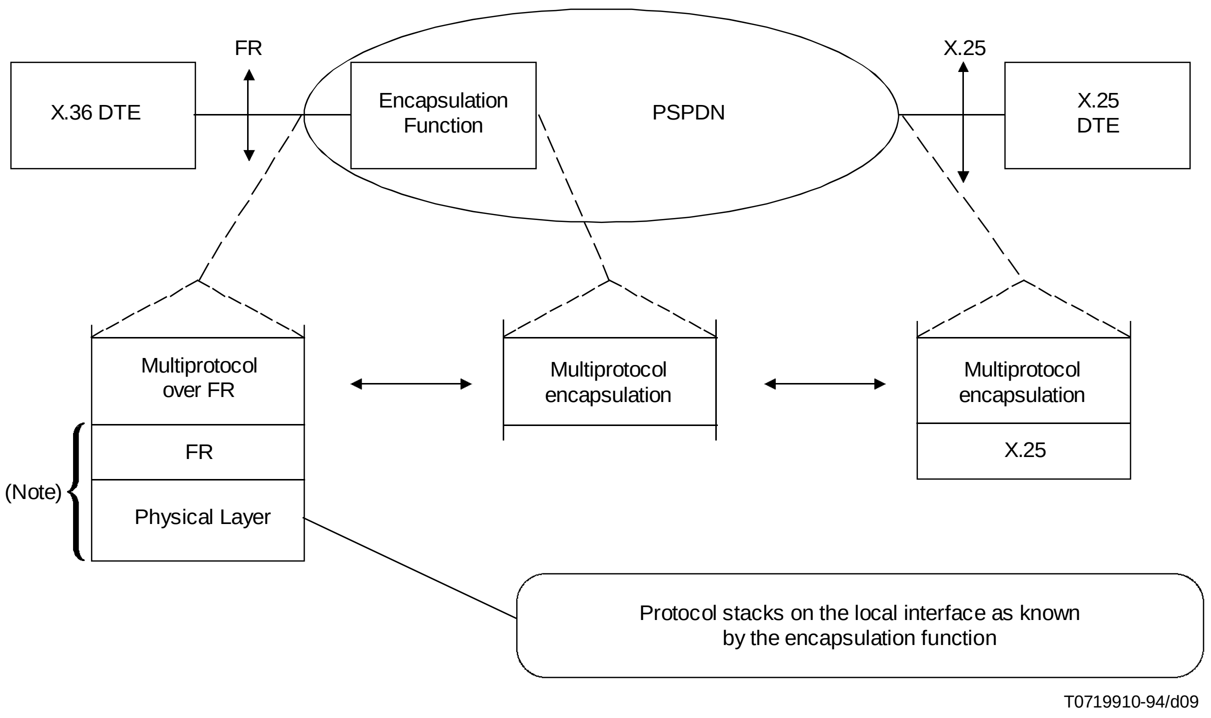
The X.25 DTE answers then with called user data specifying ISO CLNP encapsulation.

The remarks on FR procedures by the encapsulation upon flow control from the X.25 virtual call stated for the example 1 are also applicable.

X.25 DTEs which encapsulate connectionless protocols, usually clear the X.25 call after a given period of inactivity. The encapsulation function may detect this working mode through the interpretation of the clearing cause and diagnostic (see Annex B). In such a case, the concerned FR virtual circuit will still be considered as Active by the encapsulation function when answering to status enquiry messages sent by the X.36 DTE (see 9.2.3).

I.3 X.25 DTE / X.36 DTE interworking for multi-protocol encapsulation

The identified need is interworking for multi-protocol encapsulation of an X.25 DTE and an X.36 DTE connected to the same PDN (see Figure I.3).



FR
X.25
X.36 DTE
Encapsulation Function
PSPDN
X.25 DTE
Multiprotocol over FR
FR
Physical Layer
(Note)
Multiprotocol encapsulation
Multiprotocol encapsulation
X.25
Protocol stacks on the local interface as known by the encapsulation function
T0719910-94/d09

NOTE – These protocols are terminated by the encapsulation function.

FIGURE I.3/X.37

Interworking of X.25 and X.36 DTEs for multi-protocol encapsulation

The X.36 DTE uses the “multi-protocol over FR” mechanism (see clause 13/X.36).

The X.25 DTE uses the multi-protocol encapsulation mechanism described in 8.2.2.

To achieve the interworking between the X.25 and X.36 DTEs, the encapsulation function tries to set-up an X.25 virtual call towards X.25 DTE with call user data specifying multi-protocol encapsulation.

The X.25 DTE answers then with called user data specifying multi-protocol encapsulation.

The remarks on FR procedures by the encapsulation upon flow control from the X.25 virtual call stated for example 1 are also applicable.

The interworking is made easier for the encapsulation function since protocol identifier codepoints are the same.

Appendix II

Technical discussion

(This appendix does not form an integral part of this Recommendation)

II.1 Mode of operation for the multi-PDU transfer

The main goal of the multi-PDU transfer is to utilise large packets so as to optimise the end to end throughput. However, the use of the multi-PDU transfer procedure may lead to an increase in the delay before PDUs are transmitted.

To prevent from unnecessary transmission delays, when the network and/or the corresponding party are lightly loaded, the gathering of more than one encapsulated PDU in a complete packet sequence may be triggered upon flow control from the network side. Flow control on an X.25 virtual call from the network side may be detected by the encapsulation function when comparing to a predetermined threshold, the number of PDUs waiting transmission and which are in a queue within the encapsulation function.

II.2 Flow control

This subclause describes a possible implementation and shows how different flow control levels are detected and corresponding actions are undertaken.

II.2.1 Flow control from the X.25 virtual call

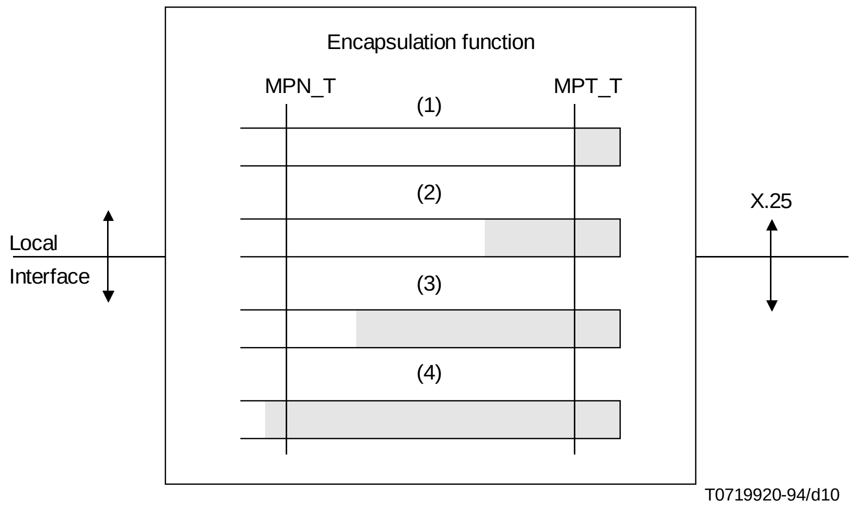
As Figure II.1 illustrates, for each X.25 virtual call, a queue contains PDUs waiting transmission.

Upon flow control on an X.25 virtual call from the PSPDN side, the number of PDUs, which are stored in the encapsulation function, waiting transmission on this X.25 virtual call, increases.

If the Multi-PDU transfer is enabled, when the number of PDUs waiting transmission on a given X.25 virtual call exceeds a predetermined threshold, called here multi-PDU transfer threshold (MPT_T), the encapsulation function begins gathering. This means that the encapsulation function sends more than one encapsulated PDUs per complete packet sequence in order to send X.25 data packets as longer as possible. Thus, the available window is efficiently used.

The MPT_T threshold value may be either global for an encapsulation function or significant per each configured X.25 virtual call, depending on the implementation. If the length of the PDUs is inferior to the data packet size, the multi-PDU transfer threshold value may correspond to the value of the window used on the X.25 virtual call.

If the number of PDUs waiting transmission is still increasing, the encapsulation function may send on the local interface some particular messages or signals according to the protocol stack used, to ask the connected equipment to reduce the number of PDUs it transmits. If these messages or signals have a global meaning for the local interface, it may not be satisfactory, since the encapsulation function may observe a strong flow control on an X.25 virtual call, and normal operation on another.



(1) For this first X.25 virtual call, there is no flow control on the X.25 virtual call from the network side and thus no gathering of PDUs is performed by the encapsulation function.

(2) There is flow control on the X.25 virtual call from the network side as detected by the multi-PDU transfer threshold (MPT_T) being exceeded. The encapsulation function gathers several PDUs in a complete packet sequence for transfer on the X.25 virtual call.

(3) There is flow control on the X.25 virtual call from the network side and the number of queued PDUs is still increasing beyond the threshold MPT_T. If possible, useful and non-disturbing to the transfer of other PDUs, a flow control message may be sent on the local interface.

(4) All long as the maximum PDU number threshold (MPN_T) is exceeded, no more PDU will be queued.

FIGURE II.1/X.37

Exemple of flow control from the X.25 virtual call

In all cases, the number of PDUs stored in the encapsulation function and waiting transmission on a given X.25 virtual call will reach a maximum limit, called here maximum PDU number threshold (MPN_T). If so, every new PDU received from the local interface and which is candidate for transmission on this X.25 virtual call, is discarded. The MPN_T threshold value may be either global for an encapsulation function or significant per each configured X.25 virtual call, depending on the implementation.

II.2.2 Flow control from the local interface side

The detection by an encapsulation function of flow control from the local interface could be based on:

- increase of the number of PDUs which are stored in the encapsulation function, waiting transmission on the local interface;
- explicit flow control messages received from the local interface.

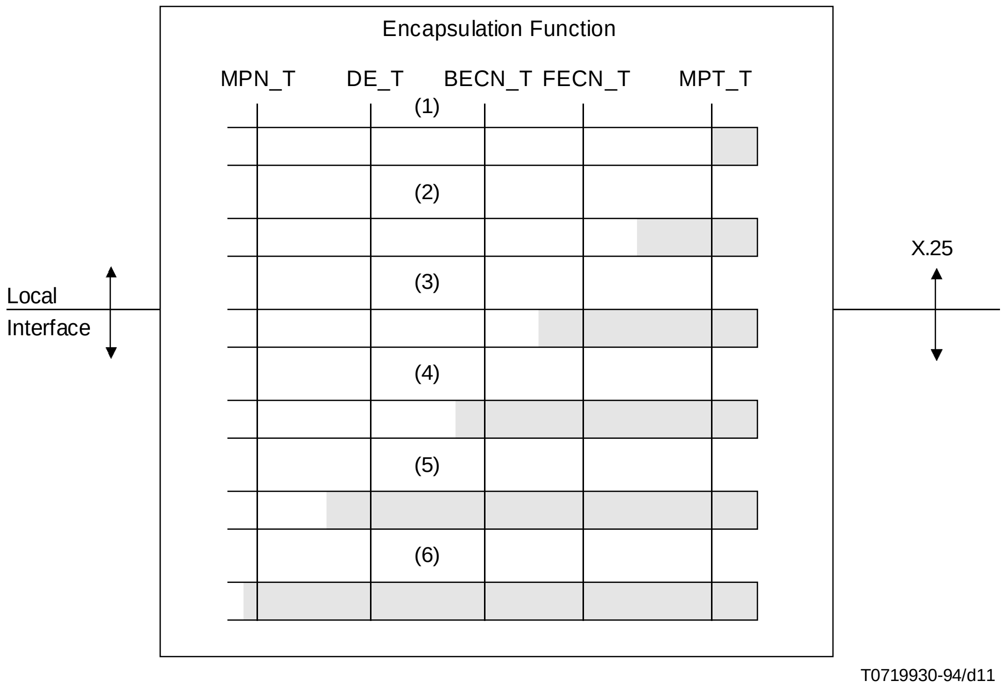
The encapsulation function may then reduce the number of PDUs it sends on the local interface possibly on a global basis or if the encapsulation function has sufficient knowledge, reduce the number of PDUs which are really affected by the flow control. The encapsulation function may also impact this flow control on the X.25 side by reducing and even closing the windows of X.25 virtual calls.

II.3 Flow control for the FR case

This subclause describes a possible implementation and shows how different flow control levels are detected and corresponding actions undertaken when FR frames are encapsulated.

II.3.1 Flow control from the X.25 virtual call

As Figure II.2 illustrates, for each X.25 virtual call, a queue contains FR frames waiting transmission.



(1) For this first X.25 virtual call, there is no flow control on the X.25 virtual call from the network side and thus no gathering of PDUs is performed by the encapsulation function.

(2) There is flow control on the X.25 virtual call from the network side as detected by the multi-PDU transfer threshold (MPT_T) being exceeded. The encapsulation function gathers several PDUs in a complete packet sequence for transfer on the X.25 virtual call.

(3) There is flow control on the X.25 virtual call from the network side and the forward explicit congestion notification threshold (FECN_T) is exceeded. The encapsulation function sets to one the FECN bit in the delimiter of FR frames from FR virtual circuit (DLCI = 69) that are encapsulated over the X.25 virtual call (LCN = 45).

(4) There is flow control on the X.25 virtual call from the network side and the backward explicit congestion noti fication threshold (BECN_T) is exceeded. The encapsulation function sets to one the BECN bit in the FR frames coming from the X.25 virtual call (LCN = 15) before sending them on the FR virtual circuit (DLCI = 56).

(5) There is flow control on the X.25 virtual call from the network side and the discard eligibility threshold (DE_T) is exceeded. Every new FR frame received from the FR interface on FR virtual circuit (DLCI = 234) which has its DE bit set to one is discarded.

(6) As long as the Maximum PDU number threshold (MPN_T) is exceeded, no more FR frame received from the FR interface on FR virtual circuit (DLCI = 555) will be queued.

FIGURE II.2/X.37

Flow control from the X.25 side and interactions on FR procedures

When the number of FR frames waiting encapsulation in a given X.25 virtual call exceeds the multi-PDU transfer threshold, the encapsulation function sends more than one FR frames per complete packet sequence since the multi-PDU transfer has been negotiated at call set-up.

When the number of FR frames stored in the encapsulation function and waiting encapsulation in a given X.25 virtual call, reaches the maximum PDU number threshold, every new FR frame received from the FR interface and which is candidate for transmission on this X.25 virtual call, is discarded.

For Frame Relay encapsulation, three additional thresholds are defined:

- Forward explicit congestion notification threshold (FECN_T);
- Backward explicit congestion notification threshold (BECN_T);
- Discard eligibility threshold (DE_T).

When the number of FR frames waiting encapsulation in a given X.25 virtual call exceeds the forward explicit congestion notification threshold, the encapsulation function sets to one the FECN bit in the delimiter of FR frames that are encapsulated in X.25 data packets.

When the number of FR frames waiting encapsulation in a given X.25 virtual call exceeds the backward explicit congestion notification threshold, the encapsulation function sets to one the BECN bit in the FR frames coming from the given X.25 virtual call, just before transmission on the FR interface.

When the number of FR frames waiting encapsulation in a given X.25 virtual call exceeds the discard eligibility threshold, every new FR frame received from the Frame Relay Interface and which is candidate for transmission on this X.25 virtual call and which has its DE bit set to one, is discarded.

The values as well as the order of the Frame Relay thresholds defined above, are implementation dependent and may be either global to an encapsulation function or significant per each configured X.25 virtual call.

II.3.2 Flow control from the local interface side

The detection by an encapsulation function of flow control from the FR interface could be based on:

- increase of the number of PDUs which are stored in the encapsulation function, waiting transmission on the FR interface;
- reception of FR frames with BECN bit set to one on given FR virtual circuits;
- etc.

The encapsulation function may then reduce the number of FR frames it sends on the affected FR virtual circuits and may also impact this flow control on the X.25 side by reducing and even closing the windows of corresponding established X.25 virtual calls.

NOTE – If the encapsulation function detect flow control from the FR interface based on the first criterion, the encapsulation function may then set to one the FECN bit in the FR frames it transmits on the FR interface and the BECN bit in the FR frames it encapsulates within X.25.

Appendix III

Protocol identification mechanism for data packets when the Q-bit is used

(This appendix does not form an integral part of this Recommendation)

Recommendation X.25 | ISO/IEC 8208 provide an optional mechanism to differentiate between user data and control information being sent while in data transfer state. This mechanism is used with the user data field to distinguish between the two types of information.

To use this mechanism, an indicator in the data packet header called the Qualifier bit (Q-bit) is used (see 4.3.6/X.25). When used, the transmitting DTE sets the Q-bit so as to have the same value (i.e. 0 or 1) in all data packets of the same complete packet sequence. The complete packet sequence will be delivered to the distant DTE as a complete packet sequence having the Q-bit set in all packets to the value assigned by the transmitting DTE.

When used, the information distinguished by the Q-bit sequences is handled based upon the procedures outlined in the relevant Recommendation or Standard. The procedures which utilise Q-bit sequences have certain common characteristics. Those known are presented here with the goal of assisting protocol developers and simplifying network management and test equipment.

Current uses of the Q-bit sequences operate under the assumption of exclusivity. That is, only one application of the Q-bit sequence is in effect on the specific virtual circuit. It has been considered that some future implementations might desire concurrent operation with the existing protocols; sharing the use of the Q-bit sequences over the same virtual circuit. This could cause some confusion.

At this point, all existing uses of the Q-bit mechanism implement it with the encoding of the first octet of the complete packet sequence. This first octet is known as the control identifier field (see 4.4.1/X.29). Known standardisation of the use of the Q-bit mechanism and the related first octets are listed in Table III.1.

The second octet is used in the ITU Recommendations to transmit additional information specific to the message type. This may require one or more additional octets (see 4.4.3/X.29 and 4.4.3/X.39). Annex A of ISO/IEC 8878 utilises the second octet to signal the message type.

TABLE III.1/X.37

Q-bit control identifier standardisation

Application	First Octet										
	Bits	8	7	6	5		4	3	2	1	
ISO/IEC											
8878, Annex A (OSI CONS over X.25) (1980 and before)		0	0	1	0		0	0	0	0	
ITU (Note 1)											
PAD Control:											
X.29 PAD		0	0	0	0		X	X	X	X	(Note 2)
X.39 PAD		0	0	0	1		X	X	X		(Note 2)
Reserved for additional PADs		0	0	1	0						
Reserved for additional PADs		0	0	1	1						
Service Extension:											
Telematic Service		0	1	0	0		X	X	X	X	(Note 2)
Reserved for additional services		0	1	0	1						
Reserved for additional services		0	1	1	0						
Reserved for additional services		0	1	1	1						
Private Extensions:											
Reserved for Private Use		1	0	0	0						
All values (inclusive) above											

NOTES

1 The extension of the control identifier field is subject to further study as presented in Note 2 of 4.4.1/X.29.

2 Bits 4, 3, 2 and 1 are used in the ITU procedures to signal the message types (see Tables 2/X.29 and 2/X.39).