This electronic version (PDF) was scanned by the International Telecommunication Union (ITU) Library & Archives Service from an original paper document in the ITU Library & Archives collections.

La présente version électronique (PDF) a été numérisée par le Service de la bibliothèque et des archives de l'Union internationale des télécommunications (UIT) à partir d'un document papier original des collections de ce service.

Esta versión electrónica (PDF) ha sido escaneada por el Servicio de Biblioteca y Archivos de la Unión Internacional de Telecomunicaciones (UIT) a partir de un documento impreso original de las colecciones del Servicio de Biblioteca y Archivos de la UIT.

هذه النسخة الإلكترونية (PDF) نتاج تصوير بالمسح الضوئي أجراه قسم المكتبة والمحفوظات في الاتحاد الدولي للاتصالات (ITU)
نقلاً من وثيقة ورقية أصلية ضمن الوثائق المتوفرة في قسم المكتبة والمحفوظات.

此电子版（PDF版本）由国际电信联盟（ITU）图书馆和档案室利用存于该处的纸质文件扫描提供。

Настоящий электронный вариант (PDF) был подготовлен в библиотечно-архивной службе Международного союза электросвязи путем сканирования исходного документа в бумажной форме из библиотечно-архивной службы МСЭ.

© International Telecommunication Union

INTERNATIONAL TELECOMMUNICATION UNION

CCITT

THE INTERNATIONAL
TELEGRAPH AND TELEPHONE
CONSULTATIVE COMMITTEE

BLUE BOOK

VOLUME VIII – FASCICLE VIII.5

DATA COMMUNICATION NETWORKS

OPEN SYSTEMS INTERCONNECTION (OSI) PROTOCOL SPECIFICATIONS, CONFORMANCE TESTING

RECOMMENDATIONS X.220-X.290

IXTH PLENARY ASSEMBLY
MELBOURNE, 14-25 NOVEMBER 1988

Geneva 1989

INTERNATIONAL TELECOMMUNICATION UNION

CCITT

THE INTERNATIONAL
TELEGRAPH AND TELEPHONE
CONSULTATIVE COMMITTEE

BLUE BOOK

VOLUME VIII – FASCICLE VIII.5

DATA COMMUNICATION NETWORKS

OPEN SYSTEMS INTERCONNECTION (OSI) PROTOCOL SPECIFICATIONS, CONFORMANCE TESTING

RECOMMENDATIONS X.220-X.290

IXTH PLENARY ASSEMBLY
MELBOURNE, 14-25 NOVEMBER 1988

Geneva 1989

ISBN 92-61-03701-1

© ITU

Printed in Belgium

CONTENTS OF THE CCITT BOOK
APPLICABLE AFTER THE NINTH PLENARY ASSEMBLY (1988)

BLUE BOOK

Volume I

FASCICLE I.1 – Minutes and reports of the Plenary Assembly.

List of Study Groups and Questions under study.

FASCICLE I.2 – Opinions and Resolutions.

Recommendations on the organization and working procedures of CCITT (Series A).

FASCICLE I.3 – Terms and definitions. Abbreviations and acronyms. Recommendations on means of expression (Series B) and General telecommunications statistics (Series C).

FASCICLE I.4 – Index of Blue Book.

Volume II

FASCICLE II.1 – General tariff principles – Charging and accounting in international telecommunications services. Series D Recommendations (Study Group III).

FASCICLE II.2 – Telephone network and ISDN – Operation, numbering, routing and mobile service. Recommendations E.100-E.333 (Study Group II).

FASCICLE II.3 – Telephone network and ISDN – Quality of service, network management and traffic engineering. Recommendations E.401-E.880 (Study Group II).

FASCICLE II.4 – Telegraph and mobile services – Operations and quality of service. Recommendations F.1-F.140 (Study Group I).

FASCICLE II.5 – Telematic, data transmission and teleconference services – Operations and quality of service. Recommendations F.160-F.353, F.600, F.601, F.710-F.730 (Study Group I).

FASCICLE II.6 – Message handling and directory services – Operations and definition of service. Recommendations F.400-F.422, F.500 (Study Group I).

Volume III

FASCICLE III.1 – General characteristics of international telephone connections and circuits. Recommendations G.100-G.181 (Study Groups XII and XV).

FASCICLE III.2 – International analogue carrier systems. Recommendations G.211-G.544 (Study Group XV).

FASCICLE III.3 – Transmission media – Characteristics. Recommendations G.601-G.654 (Study Group XV).

FASCICLE III.4 – General aspects of digital transmission systems; terminal equipments. Recommendations G.700-G.795 (Study Groups XV and XVIII).

FASCICLE III.5 – Digital networks, digital sections and digital line systems. Recommendations G.801-G.961 (Study Groups XV and XVIII).

FASCICLE III.6 – Line transmission of non-telephone signals. Transmission of sound-programme and television signals. Series H and J Recommendations (Study Group XV).

FASCICLE III.7 – Integrated Services Digital Network (ISDN) – General structure and service capabilities. Recommendations I.110-I.257 (Study Group XVIII).

FASCICLE III.8 – Integrated Services Digital Network (ISDN) – Overall network aspects and functions, ISDN user-network interfaces. Recommendations I.310-I.470 (Study Group XVIII).

FASCICLE III.9 – Integrated Services Digital Network (ISDN) – Internetwork interfaces and maintenance principles. Recommendations I.500-I.605 (Study Group XVIII).

Volume IV

FASCICLE IV.1 – General maintenance principles: maintenance of international transmission systems and telephone circuits. Recommendations M.10-M.782 (Study Group IV).

FASCICLE IV.2 – Maintenance of international telegraph, phototelegraph and leased circuits. Maintenance of the international public telephone network. Maintenance of maritime satellite and data transmission systems. Recommendations M.800-M.1375 (Study Group IV).

FASCICLE IV.3 – Maintenance of international sound-programme and television transmission circuits. Series N Recommendations (Study Group IV).

FASCICLE IV.4 – Specifications for measuring equipment. Series O Recommendations (Study Group IV).

Volume V – Telephone transmission quality. Series P Recommendations (Study Group XII).

Volume VI

FASCICLE VI.1 – General Recommendations on telephone switching and signalling. Functions and information flows for services in the ISDN. Supplements. Recommendations Q.1-Q.118 *bis* (Study Group XI).

FASCICLE VI.2 – Specifications of Signalling Systems Nos. 4 and 5. Recommendations Q.120-Q.180 (Study Group XI).

FASCICLE VI.3 – Specifications of Signalling System No. 6. Recommendations Q.251-Q.300 (Study Group XI).

FASCICLE VI.4 – Specifications of Signalling Systems R1 and R2. Recommendations Q.310-Q.490 (Study Group XI).

FASCICLE VI.5 – Digital local, transit, combined and international exchanges in integrated digital networks and mixed analogue-digital networks. Supplements. Recommendations Q.500-Q.554 (Study Group XI).

FASCICLE VI.6 – Interworking of signalling systems. Recommendations Q.601-Q.699 (Study Group XI).

FASCICLE VI.7 – Specifications of Signalling System No. 7. Recommendations Q.700-Q.716 (Study Group XI).

FASCICLE VI.8 – Specifications of Signalling System No. 7. Recommendations Q.721-Q.766 (Study Group XI).

FASCICLE VI.9 – Specifications of Signalling System No. 7. Recommendations Q.771-Q.795 (Study Group XI).

FASCICLE VI.10 – Digital subscriber signalling system No. 1 (DSS 1), data link layer. Recommendations Q.920-Q.921 (Study Group XI).

FASCICLE VI.11 – Digital subscriber signalling system No. 1 (DSS 1), network layer, user-network management. Recommendations Q.930-Q.940 (Study Group XI).

FASCICLE VI.12 – Public land mobile network. Interworking with ISDN and PSTN. Recommendations Q.1000-Q.1032 (Study Group XI).

FASCICLE VI.13 – Public land mobile network. Mobile application part and interfaces. Recommendations Q.1051-Q.1063 (Study Group XI).

FASCICLE VI.14 – Interworking with satellite mobile systems. Recommendations Q.1100-Q.1152 (Study Group XI).

Volume VII

FASCICLE VII.1 – Telegraph transmission. Series R Recommendations. Telegraph services terminal equipment. Series S Recommendations (Study Group IX).

FASCICLE VII.2 – Telegraph switching. Series U Recommendations (Study Group IX).

FASCICLE VII.3 – Terminal equipment and protocols for telematic services. Recommendations T.0-T.63 (Study Group VIII).

FASCICLE VII.4 – Conformance testing procedures for the Teletex Recommendations. Recommendation T.64 (Study Group VIII).

FASCICLE VII.5 – Terminal equipment and protocols for telematic services. Recommendations T.65-T.101, T.150-T.390 (Study Group VIII).

FASCICLE VII.6 – Terminal equipment and protocols for telematic services. Recommendations T.400-T.418 (Study Group VIII).

FASCICLE VII.7 – Terminal equipment and protocols for telematic services. Recommendations T.431-T.564 (Study Group VIII).

Volume VIII

FASCICLE VIII.1 – Data communication over the telephone network. Series V Recommendations (Study Group XVII).

FASCICLE VIII.2 – Data communication networks: services and facilities, interfaces. Recommendations X.1-X.32 (Study Group VII).

FASCICLE VIII.3 – Data communication networks: transmission, signalling and switching, network aspects, maintenance and administrative arrangements. Recommendations X.40-X.181 (Study Group VII).

FASCICLE VIII.4 – Data communication networks: Open Systems Interconnection (OSI) – Model and notation, service definition. Recommendations X.200-X.219 (Study Group VII).

FASCICLE VIII.5 – Data communication networks: Open Systems Interconnection (OSI) – Protocol specifications, conformance testing. Recommendations X.220-X.290 (Study Group VII).

FASCICLE VIII.6 – Data communication networks: interworking between networks, mobile data transmission systems, internetwork management. Recommendations X.300-X.370 (Study Group VII).

FASCICLE VIII.7 – Data communication networks: message handling systems. Recommendations X.400-X.420 (Study Group VII).

FASCICLE VIII.8 – Data communication networks: directory. Recommendations X.500-X.521 (Study Group VII).

Volume IX – Protection against interference. Series K Recommendations (Study Group V). Construction, installation and protection of cable and other elements of outside plant. Series L Recommendations (Study Group VI).

Volume X

CONTENTS OF FASCICLE VIII.5 OF THE BLUE BOOK

Recommendations X.220 to X.290

Open Systems Interconnection (OSI)

PRELIMINARY NOTES

1 The Questions entrusted to each Study Group for the Study Period 1989-1992 can be found in Contribution No. 1 to that Study Group.

2 In this Fascicle, the expression "Administration" is used for shortness to indicate both a telecommunication Administration and a recognized private operating agency.

3 The status of annexes and appendices attached to the Series X Recommendations should be interpreted as follows (except where specified):

– an *annex* to a Recommendation forms an integral part of Recommendation;

– an *appendix* to a Recommendation does not form part of the Recommendation and only provides some complementary explanation or information specific to that Recommendation.

4 A number of the Series X Recommendations contained in this Fascicle were jointly developed in collaboration with the ISO/IEC. Cross-references between these Recommendations and the corresponding ISO/IEC standards are given in the table below.

CCITT Recommendation	ISO/IEC Standard
X.223	ISO 8878, Information processing systems – Data communications – Use of X.25 to provide the OSI connection-mode network service (1987).
X.224	ISO 8073, Information processing systems – Open Systems Interconnection – Connection oriented transport protocol specification (1986).
X.225	ISO 8327, Information processing systems – Open Systems Interconnection – Basic connection oriented session protocol specification (1987).
	ISO 8327/AD2, Information processing systems – Open Systems Interconnection – Basic connection oriented session protocol specification – Addendum 2: Incorporation of unlimited user data [a].
X.226	ISO 8823, Information processing systems – Open Systems Interconnection – Connection oriented presentation protocol specification (1988).
X.227	ISO 8650, Information processing systems – Open Systems Interconnection – Protocol specification for the Association Control Service Elements [b].
X.228	ISO 9066-2, Information processing systems – Text communication – Reliable Transfer – Part 2: Protocol specification [c].
X.229	ISO 9072-2, Information processing systems – Text communication – Remote Operations – Part 2: Protocol specification [c].
X.290	ISO 9646-1, Information processing systems – Open Systems Interconnection – OSI conformance testing methodology and framework – Part 1: General concepts [d].
	ISO 9646-2, Information processing systems – Open Systems Interconnection – OSI conformance testing methodology and framework – Part 1: Abstract test suite specification [d].

[a] Presently at the stage of Draft Addendum (DAD).

[b] Presently awaiting publication.

[c] Presently at the stage of Draft International Standard (DIS).

[d] Presently at the stage of Draft Proposal (DP).

FASCICLE VIII.5

Recommendations X.220 to X.290

DATA COMMUNICATION NETWORKS: OPEN SYSTEMS INTERCONNECTION (OSI); PROTOCOL SPECIFICATIONS, CONFORMANCE TESTING

PAGE INTENTIONALLY LEFT BLANK

PAGE LAISSEE EN BLANC INTENTIONNELLEMENT

SECTION 3

PROTOCOL SPECIFICATIONS

Recommendation X.220

USE OF X.200-SERIES PROTOCOLS IN CCITT APPLICATIONS

(Melbourne, 1988)

The CCITT,

considering

(a) that Administrations in many countries are implementing a variety of telecommunications services;

(b) that these services may be carried on a variety of networks;

(c) that users of these services desire a unifying architecture for the applicable protocols;

(d) that such an architecture is provided by Recommendation X.200 which defines the Reference Model of Open Systems Interconnection for CCITT Applications;

(e) that a number of protocols conforming to this architecture are defined in the X.200 Series of Recommendations and in other Recommendations,

unanimously declares

that for CCITT Applications, the functional suites of protocols, which involve the use of protocols in the X.200 series of Recommendations, are summarized in this Recommendation. Details, as well as any conformance requirements, are contained in the relevant Recommendations.

A growing number of data terminal equipments are being designed to support more than one CCITT service and/or are being designed to be capable of being connected to more than one type of network. In order to facilitate the design of such equipments, the various OSI protocol suites involving use of the X.200 series of Recommendations are documented herein.

These protocol suites are depicted in Figure 1/X.220, which portrays the protocols according to the seven layers defined in Recommendation X.200. The CCITT Applications covered are Message Handling Systems (MHS), Directory, Teletex and Document Architecture Transfer and Manipulation. The networks covered are PSPDN, CSPDN, PSTN and ISDN. The intent is to give a general view of the set of protocol suites in a single figure, while relying on the other Recommendations referenced to provide the necessary additional details.

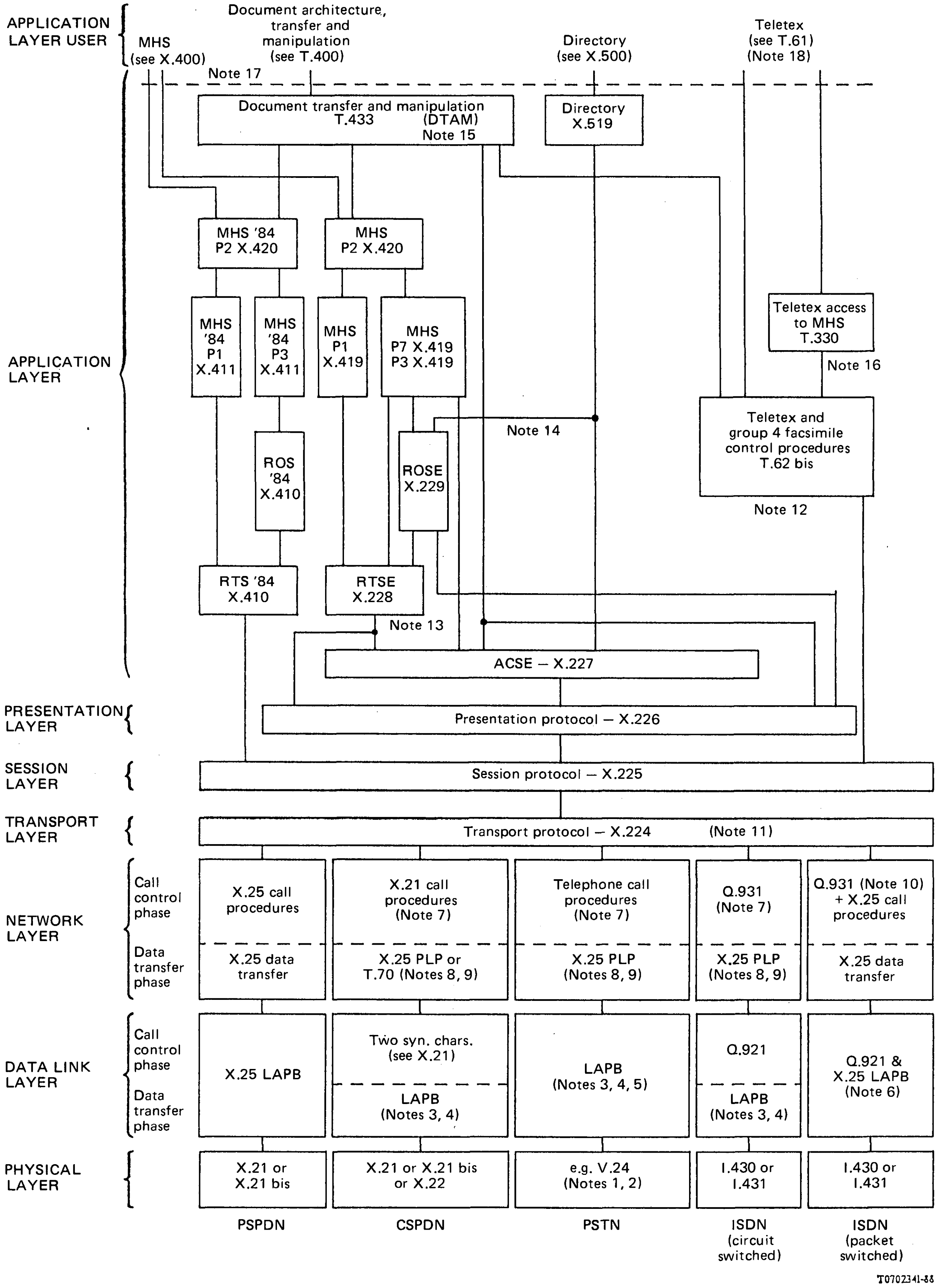


FIGURE 1/X.220

Protocol suites

Notes for Figure 1/X.220

Note 1 – The modem may also be integrated within the terminal and in such cases V.24 need not apply. For telematic terminals, see § 3.2.1 of T.70.

Note 2 – For automatic calling and/or answering, V.25 or V.25 *bis* may be applicable.

Note 3 – For terminals connected to a PSTN, CSPDN or ISDN (circuit switched) and accessing a PSPDN in accordance with X.32 or X.31, the X.25 LAPB procedures are used as set forth in X.32 or X.31.

Note 4 – For DTE-to-DTE connections, telematic terminals employ the X.75 LAPB procedures for single link operation (see § § 3.2.2 and 3.3.2 of T.70 and § 2.1.2.2 of T.90). For other terminals, the ISO 7776 LAPB procedures may apply for DTE-to-DTE connections.

Note 5 – For half duplex operation over the PSTN, the LAPB procedures are extended to include a half duplex transmission module (HDTM) as defined in § 5.6 of X.32 and in T.71.

Note 6 – Terminals obtaining packet access on the D-channel use the LAPD procedures of Q.921 to support both the Q.931 access connection control procedure (if needed) and the X.25 packet layer procedures. Terminals obtaining packet access on the B-channel use the LAPD procedures of Q.921 to support the Q.931 access connection control procedure (if needed) and the X.25 LAPB procedures to support the X.25 packet layer procedures.

Note 7 – For terminals connected to a PSTN, CSPDN or ISDN (circuit switched) and accessing a PSPDN in accordance with X.32 or X.31, the network connection is established by two stage selection; the first stage uses the call control procedures of the attached network (as shown in Figure 1/X.220) and the second stage uses the X.25 call control procedures.

Note 8 – For terminals connected to a PSTN, CSPDN or ISDN (circuit switched) and accessing a PSPDN in accordance with X.32 or X.31, the X.25 packet layer procedures apply during the data transfer phase of the PSTN, CSPDN or ISDN. However, for telematic terminals connected to a CSPDN and accessing a PSPDN, a minimum network layer functionality is required during the data transfer phase of the CSPDN (see § 3.3.3 of T.70).

Note 9 – For DTE-to-DTE connections, telematic terminals connected to a CSPDN use the minimum network layer functionality (see § 3.3.3 of T.70) during the data transfer phase of the CSPDN and telematic terminals connected to a PSTN use the X.25 packet layer procedures (see § 3.2.3 of T.70); telematic terminals connected to an ISDN (circuit switched) use the X.25 packet layer procedures as specified in ISO 8208 (see § 2.1.2.3.2 of T.90) or, in addition as a user option, the minimum network layer functionality (see § 2.1.1 of T.90). For other terminals, the ISO 8208 X.25 packet layer procedures may apply for DTE-to-DTE connections.

Note 10 – The Q.931 access connection control procedures are used if needed.

Note 11 – For telematic terminals, the transport protocol is in accordance with T.70 Section 5 plus Annexes A and B; the use of X.224 class 0 plus application rules is optional but needs further consideration to ensure that there are no discrepancies with T.70. For terminals communicating with network based services such as MHS and Directories, the X.224 procedures apply including the mandatory support of class 0.

Note 12 – T.62 *bis,* together with the relevant service and protocol elements of X.215 and X.225, are intended to be equal to T.62.

Note 13 – To obtain backward compatibility with X.410 (1984), RTSE uses the "X.410-1984 mode" services of ACSE and the Presentation Layer. The "normal mode" is used in all other cases.

Note 14 – Directory uses ROSE but not RTSE.

Note 15 – The use of ROSE and RTSE in the T.400-series is for further study.

Note 16 – T.330 describes Group 4 Facsimile and Teletex access to the MHS Interpersonal Messaging System (IPMS) in the T.62 *bis*/X.225 environment.

Note 17 – The use of MHS to transfer documents conforming to the T.410-series is described in T.411.

Note 18 – Character repertoire definition of T.61 only.

Recommendation X.223

USE OF X.25 TO PROVIDE THE OSI CONNECTION-MODE NETWORK SERVICE FOR CCITT APPLICATIONS

(Melbourne, 1988)

The CCITT,

considering

(a) that Recommendation X.200 defines the Reference Model of Open Systems Interconnection for CCITT Applications;

(b) that Recommendation X.210 gives the Open Systems Interconnection (OSI) Layer Service Definition Conventions;

(c) that Recommendation X.213 is the Network Service Definition for Open Systems Interconnection for CCITT Applications;

(d) that Recommendation X.25 describes the 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 Circuit;

(e) that Recommendation X.96 defines the Call Progress Signals in Public Data Networks,

unanimously declares

(1) that the scope, field of application, references definitions, symbols and abbreviations are given in §§ 1 to 4;

(2) that the mapping of X.25 protocol elements to CONS primitives is overviewed in § 5;

(3) that the detailed mapping for the network connection establishment phase described in § 6;

(4) that the detailed mapping for the network connection release phase is described in § 7;

(5) that the detailed mapping for the data transfer phase is described in §§ 8 to 11.

CONTENTS

0 Introduction

This Recommendation defines the method for providing the OSI Connection-Mode Network Service (CONS) for CCITT Applications through the use of the X.25 Packet Layer Protocol (X.25/PLP). Particularly, it specifies a mapping between the elements of the X.25/PLP and the primitives of the OSI CONS specified in Recommendation X.213.

Appendix I provides additional considerations on the relationship between the X.25 protocol procedures and the CONS primitives.

Appendix II illustrates the use of X.25 Network Protocol Address Information (NPAI), i.e., the Address Field and the Address Extension Facilities.

Appendix III illustrates the use of X.25 transit delay facilities.

The relationship between the X.25/PLP and the OSI CONS is shown in Figure 1/X.223. This relationship is described only in terms of the Network Layer entities that provide the CONS. No discussion is given here to describe the actions of a network layer entity that provides only a relay function for a given network connection.

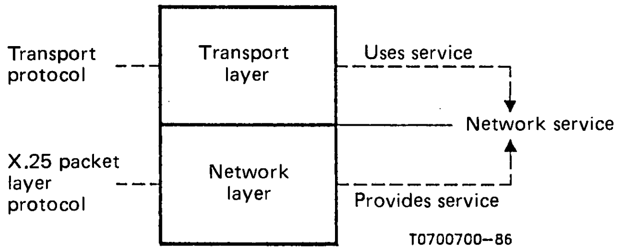


FIGURE 1/X.223

Relationship of the X.25 Packet Layer Protocol to the OSI Connection-Mode Network Service

The OSI Network Service is defined in terms of:

a) the primitive actions and events of the Service;

b) the parameters associated with each primitive action and event, and the form which they take; and

c) the interrelationship between, and the valid sequences of, these actions and events.

The OSI Network Service does not specify individual implementations or products nor does it constrain the implementation of entities and interfaces within a computer system.

The X.25/PLP is defined in terms of:

a) procedures for Virtual Calls and Permanent Virtual Circuits;

b) formats for packets assocaiated with these procedures; and

c) procedures and formats for optional user facilities and CCITT-Specified DTE facilities.

The use of the word "Network" to name the "Network" Layer of the OSI Reference Model should be distinguished from the use of the word "network" to denote a communications network as conventionally understood. To facilitate this distinction, the term "subnetwork" is used for a collection of physical equipment, commonly called a "network" (reference Recommendation X.200). Subnetworks may be either public or private networks. In the case of public networks, their properties may be determined by separate Recommendations such as X.21 for a circuit-switched network or X.25 for a packet-switched network.

Throughout the set of OSI standards, the term "Service" refers to the abstract capability provided by one layer of the OSI Reference Model to the layer above it. Thus, the Network Service is a conceptual architectural Service, independent of administrative divisions.

Note – It is important to distinguish the specialized use of the term "Service" within the set of OSI standards from its use elsewhere to describe the provision of a service by an organization (such as the provision of a service by an Administration).

1 Scope and Field of Application

The OSI CONS, as stated above, is defined in terms of a set of primitive actions and events and associated parameters. For a protocol to support this service, there must be a mapping between the abstract primitives and parameters of the CONS and the real elements of the protocol. This Recommendation provides such a mapping for the X.25/PLP.

The X.25/PLP is usually regarded as operating between an end system (i.e., a "Data Terminal Equipment" in X.25 terminology) and a packet-switched public data subnetwork. However, the X.25/PLP can also be used in other environments to provide the OSI CONS. Examples of such other uses may be the direct connection or circuit-switched connection (including connection across a circuit-switched data subnetwork) of two end systems without an intervening packet-switched public data subnetwork, or the connection of an end system to an Integrated Service Digital Network.

Guidance for the design of DTEs is available in ISO standard ISO 8208 (see reference 4).

2 References

[1] Recommendation X.200 – Reference Model of Open Systems Interconnection for CCITT Applications.

Note – See also ISO 7498 – OSI Basic Reference Model.

[2] Recommendation X.210 – Open Systems Interconnection Layer Service Definition Conventions.

Note – See also ISO TR 8509 – Network Service Connection.

[3] Recommendation X.213 – Network Service Definition for Open Systems Interconnection for CCITT Applications.

Note – See also ISO 8348 – Network Service Definition.

[4] Recommendation X.25 – 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 Circuit.

Note – This Recommendation is referred solely with respect to its packet layer protocol description. However, this Recommendation fully specifies the behaviour of the DCE while specifying only a minimum set of requirements for the DTE. Additional guidance for the design of DTEs is available in ISO standard ISO 8208. The development of a Recommendation describing X.25 DTE procedures for CCITT Applications is for further study.

[5] Recommendation X.96 – Call Progress Signals in Public Data Networks.

[6] ISO 8878 – Information Processing Systems – Data Communications – Use of X.25/PLP to Provide the OSI Connection-Mode Network Service.

3 Definitions

3.1 *Reference Model Definitions*

The following concepts, developed and defined in the OSI Reference Model (Recommendation X.200), are used:

a) Network connection,

b) Network Layer,

c) Network Service,

d) Network Service Access Point,

e) Network Service Access Point Address,

f) Subnetwork.

3.2 *Service Conventions Definitions*

The following terms, as they apply to the Network Layer and as defined in the Service Conventions Standard (Recommendation X.210), are used:

g) Network Service user,

h) Network Service provider,

i) primitive,

j) request,
k) indication,
l) response,
m) confirm.

3.3 *Network Service Definitions*

The following terms, as defined in the Network Service (Recommendation X.213), are used:

n) Calling Network Service user,
o). Called Network Service user.

3.4 *Addressing Definitions*

The following concepts, as defined in Recommendation X.213, are used:

p) Subnetwork Point of Attachment address,
q) Network Protocol Address Information,
r) Initial Domain Part,
s) Authority and Format Identifiers,
t) Initial Domain Identifier,
u) Domain Specific Part.

3.5 *X.25 Definitions*

The following concepts, as developed in Recommendation X.25, are used:

v) Virtual Circuit,
w) Virtual Call,
x) Logical Channel,
y) Packet Layer,
z) Data Terminal Equipment,
aa) Data Circuit-terminating Equipment,
ab) DXE (either a DTE or a DCE).

3.6 *X.96 Definitions*

The following terms, as defined in Recommendation X.96, are used:

ac) Category C call progress signal,
ad) Category D call progress signal.

4 **Abbreviations**

4.1 *Network Service Abbreviations*

CONS	Connection-Mode Network Service
N	Network
NC	Network Connection
NL	Network Layer
SR	Network Service
NSAP	Network Service Access Point
OSI	Open Systems Interconnection
QOS	Quality of Service

4.2 *Addressing Abbreviations*

AFI	Authority and Format Identifier
DSP	Domain Specific Part
IDI	Initial Domain Identifier
IDP	Initial Domain Part
NPAI	Network Protocol Address Information
SNPA	Subnetwork Point of Attachment

4.3 *X.25 Abbreviations*

AEF	Address Extension Facility
AF	Address Field
D-Bit	Delivery Confirmation bit
DCE	Data-Circuit-terminating Equipment
DTE	Data Terminal Equipment
EDN	Expedited Data Negotiation (Facility)
EETDN	End-to-End Transit Delay Negotiation (Facility)
FPF	Facility Parameter Field
GFI	General Format Identifier
LC	Logical channel
M-bit	More Data bit
MBS	M-bit Sequence
MTCN	Minimum Throughput Class Negotiation (Facility)
PLP	Packet Layer Protocol
P(R)	Packet receive sequence number
P(S)	Packet send sequence number
TCN	Throughput Class Negotiation (Facility)
TDSAI	Transit Delay Selection And Indication (Facility)
VC	Virtual Call

5 Overview

This Network Service (NS) provides for the transponder transfer of data between NS users. It makes invisible to these NS users the way in which supporting communications resources are utilized to achieve this transfer.

5.1 *Elements of the X.25/PLP Used to Support the OSI CONS*

The X.25/PLP, as defined by Recommendation X.25, provides a specific realization for the transparent transfer of data between NS users of the CONS. The elements of this protocol to be considered are:

a) the virtual-circuit types;

b) the packet types and fields to be mapped to the primitives and parameters of the OSI CONS; and

c) the optional user facilities and CCITT-Specified DTE facilities.

Of the two types of virtual circuits defined in Recommendation X.25, the use of Virtual Calls (VCs) is mapped to the Network Connection (NC) Establishment and Release Phases of the OSI CONS.

Table 1/X.223 below lists the X.25/PLP packets and associated fields that shall be used when supporting the OSI CONS.

TABLE 1/X.223

Packets and Fields of the X.25/PLP used to support the OSI CONS

Packet Types [1]	Fields [2]
CALL REQUEST INCOMING CALL CALL ACCEPTED CALL CONNECTED	General Format Identifier [3], Address Field, Facility Field, Call and Called User Data Field [4]
CLEAR REQUEST CLEAR INDICATION	Clearing Cause Field, Diagnostic Code Field, Address Field, Facility Field, Clear User Data Field [4]
DATA	D-bit, M-bit, P(S) [5], P(R) [5], User Data Field [4]
INTERRUPT	Interrupt User Data Field [4]
RECEIVE READY [6] RECEIVE NOT READY [6] REJECT [6] (if agreed to)	P(R) [5]
RESET REQUEST RESET INDICATION	Resetting Cause Field, Diagnostic Code Field
RESTART INDICATION	Restarting Cause Field, Diagnostic Code Field

Note 1 – The packets shown in the table are used in support of the primitives of the OSI CONS. Other packets not shown in the table (i.e. CLEAR CONFIRMATION, INTERRUPT CONFIRMATION, RESET CONFIRMATION and RESTART CONFIRMATION packets) are essential to the use of the packets shown. Yet other packets (i.e. RESTART REQUEST, DIAGNOSTIC, REGISTRATION REQUEST, and REGISTRATION CONFIRMATION packets) have no relationship to the provision of the OSI CONS.

Note 2 – The information in the fields shown in the table have a direct relationship to the parameters associated with the primitives of the OSI CONS. Other fields not shown in the table (e.g. the Logical Channel Identifier, the Packet Type Identifier, the Address Length Fields, and the Facility Length Field) are essential to the use of the appropriate packets.

Note 3 – Bit 7 of octet 1 of the General Format Identifier (GFI) in this packets is used to negociate the overall availability of the Delivery Confirmation bit (D-bit) in support of the Receipt Confirmation Service. As such, this bit has no specific field-name as defined in the X.25/PLP.

Note 4 – All user data fields are octet aligned.

Note 5 – The P(S) and P(R) fields are essential to the operation of the X.25/PLP in providing the Receipt Confirmation Service.

Note 6 – The action implied by these packets has no relationship to the primitives of the OSI CONS. However, the P(R) field is essential to the operation of the X.25/PLP in providing the Receipt Confirmation Service.

In addition, the following optional user facilities and CCITT-Specified DTE facilities shall be used and/or agreed to:

a) optional user facilities:

- Fast Select (facility used; when operating in a DTE-to-DTE environment without an intervening packet-switched network, the use of the Fast Select Facility shall be agreed to by the two DTEs);
- Fast Select Acceptance (facility agreed to if operating in a packet-switched network environment);
- Throughput Class Negotiation (facility agreed to and used); and
- Transit Delay Selection And Indication (facility used; when operating in a DTE-to-DTE environment without an intervening packet-switched network, use of the TDSAI facility is for further study).

b) CCITT-Specified DTE facilities:

- Called Address Extension (facility used);
- Calling Address Extension (facility used);
- End-to-End Transit Delay Negotiation (facility used);
- Expedited Data Negotiation (facility used); and
- Minimum Throughput Class Negotiation (facility used).

5.2 *General Operation of the X.25/PLP For Supporting the OSI CONS*

The X.25/PLP can be used to provide the OSI CONS in an end system connected to a public X.25 packet-switched subnetwork. It can also be used in environments where end systems are connected by a dedicated path or by a circuit-switched connection.

As shown in Figure 2/X.223, the NS provider (more particularly, the Network Layer (NL) entity in an end system) must provide a translation between:

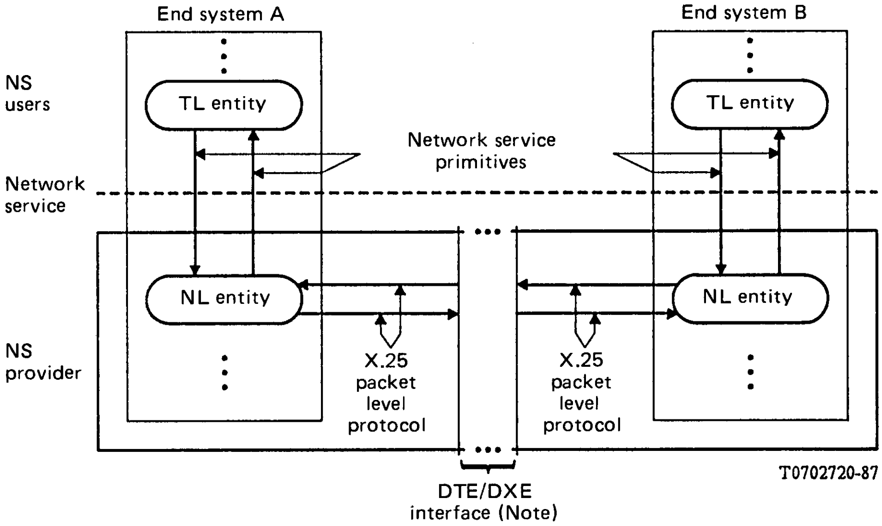
a) the primitives and parameters of the OSI CONS; and

b) the packets and associated fields of the X.25/PLP.

Request and response primitives are translated into packets to be transmitted across the DTE/DXE interface by the NL entity. Received packets, where appropriate, are translated by the NL entity into indication and confirm primitives.

Appendix I provides additional considerations on the relationship between the X.25 protocol procedures and the CONS primitives.

Note – The Network Service Definition specifies valid sequences of primitives at an NC endpoint and valid parameter responses at the called NC endpoint to Receipt Confirmation negotiation, Expedited Data negotiation, and Quality of Service (QOS) parameter negotiation. The necessity for the NL entity to monitor compliance and the actions to be taken on non-compliance are a local matter, and not subject to standardization.



Note – This interface consists of zero or more Network Layer entities providing a Network Layer relay function.

FIGURE 2/X.223

Operation of OSI Connection-Mode Network Service and X.25 Packet Layer Protocol

There is also a relationship between some local mechanism used to identify a particular NC and a Logical Channel (LC) number used to identify a particular virtual circuit. This relationship is a local matter and is not discussed here.

6 Network Connection Establishment Phase

Sections 6 to 11 of this Recommendation deal with the mapping of the OSI CONS to/from the X.25/PLP.

6.1 *Primitive/Parameter and Packet/Field Relationships*

Table 2/X.223 shows the relationships between the primitives/parameters used during the Network Connection Establishment Phase and the packets/fields associated with the Call Set-up Procedures.

6.2 *Procedures*

6.2.1 *Primitive/Packet Mapping*

When an NL entity receives an N-CONNECT request or an N-CONNECT response primitive from an NS user, it transmits a CALL REQUEST or a CALL ACCEPTED packet, respectively, across the DTE/DXE interface.

When an NL entity receives an INCOMING CALL or a CALL CONNECTED packet, it signals an N-CONNECT indication or an N-CONNECT confirm primitive, respectively, to the NS user.

6.2.2 *NSAP Addresses*

Local operation determines the contents of the Network Protocol Address Information (NPAI) and whether Network Service Access Point (NSAP) Addresses, where explicitly supplied, are mapped to and from the Address Field (AF) or the Address Extension Facilities (AEF) of X.25/PLP call set-up packets. Appendix II describes guidelines for the methods by which the required AF contents may be derived from the NSAP Address. The permitted techniques for the placement of NSAP Addresses in either the AF or AEF are given in this clause. The encoding techniques to be employed are those specified in Recommendation X.25 for the AF and AEF. The content of these fields shall be in the preferred binary encoding defined in Recommendation X.213. Examples of encoding NSAP Addresses in the NPAI of the X.25/PLP are also given in Appendix II.

Note – The use of the preferred binary encoding results in binary-coded decimal digits in the AF, as required by Recommendation X.25.

6.2.2.1 *Encoding of NSAP Addresses*

6.2.2.1.1 *Use of the Address Field (AF)*

Under certain conditions, the NSAP Address, as defined in Recommendation X.213, is conveyed entirely in the AF. These conditions are:

a) the NSAP Address consists solely of the Intial Domain Part (IDP), i.e., the Domain Specific Part (DSP) is null;

b) the Authority and Format Identifier (AFI) can be deduced from the contents of the AF (e.g., with knowledge of the subnetwork to which the DTE is attached);

c) the Initial Domain Identifier (IDI) is the same as the Subnetwork Point of Attachment (SNPA) Address; and

d) the NL entity, through local knowledge, is aware that the remote NL entity does not operate according to CCITT Recommendation X.223 and cannot recognize the AEF.

When all of the above conditions are satisfied, the AF is used to convey the semantics of the entire NSAP Address (the AFI is implied and the contents of the AF are equivalent to the IDI).

TABLE 2/X.223

CONS:X.25/PLP Mapping for the Network Connection Establishment Phase

CONS	X.25/PLP
PRIMITIVES:	PACKETS:
N-CONNECT request	CALL REQUEST
N-CONNECT indication	INCOMING CALL
N-CONNECT response	CALL ACCEPTED
N-CONNECT confirm	CALL CONNECTED
PARAMETERS:	FIELDS (INCLUDING FACILITIES):
Called Address	Called DTE Address Field Called Address Extension Facility
Calling Address	Calling DTE Address Field Calling Address Extension Facility
Responding Address	Called DTE Address Field Called Address Extension Facility
Receipt Confirmation Selection	General Format Identifier [1]
Expedited Data Selection	Expedited Data Negotiation Facility
QoS-Parameter Set	Throughput Class Negotiation Facility [2]
	Minimum Throughput Class Negotiation Facility
	Transit Delay Selection And Indication Facility
	End-to-End Transit Delay Negotiation Facility
NS-User-Data	Call and Called User Data Field
	Fast Select Facility [3]

Note 1 – Bit 7 of octet 1 of the GFI in call setup packets is used to negociate the overall availability of the D-bit in support of the Receipt Confirmation Service. As such, this bit has no specific field-name as defined in the X.25/PLP.

Note 2 – For proper operation, this optional user facility shall also be agreed to for use on the interface.

Note 3 – For proper operation, the Fast Select Acceptance Facility shall also be agreed to on the interface when accessing a packet-switched network.

6.2.2.1.2 *Use of the Address Extension Facility (AEF)*

If any of the conditions in § 6.2.2.1.1 are not satisfied, the AEF shall be used. The NSAP address, complete with the AFI, is placed in the AEF (bits 8 and 7 of the first octet of the Facility Parameter Field (FPF) are both set to zero). In this case, the contents of the AF are not defined by this Recommendation. Guidelines for their derivation are given in Appendix II.

6.2.2.2 *Decoding of the NSAP Addresses*

6.2.2.2.1 *Absent AEF Case*

If the AEF is not present, then local knowledge is required by the receiving NL entity to determine whether an OSI NAP Address is to be deduced from the content of the AF. If this local exchange indicates that an NSAP Address is present, its abstract syntax is as follows:

a) the AFI is deduced from knowledge of the subnetwork from which the packet was received;

b) the IDI is the same as the contents of the AF; and

c) the DSP is absent.

6.2.2.2.2 *AEF Case*

If the AEF is present and bits 8 and 7 of the leading octet of the FPF are both set to zero, then the NSAP Address is contained entirely within the AEF. The abstract synstax is as follows:

a) the AFI is contained within the first two digits of the AEF;

b) the IDI is the remainder of the IDP after any leading and trailing padding digits are discarded; and

c) the DSP, if present, constitutes the remainder of the AEF content after any trailing padding digits are discarded.

6.2.3 *Receipt Confirmation Selection*

Bit 7 of octet 1 in the GFI of X.25/PLP call set-up packets is mapped to/from the Receipt Confirmation Selection parameter of N-CONNECT primitives.

If the Receipt Confirmation Selection parameter of the N-CONNECT request primitives indicates "use of Receipt Confirmation", then the NL entity, if it can support the D-bit Procedure as defined in §§ 8.2.3 and 9.2.1, sets bit 7 of the GFI to 1 to indicate use of receipt confirmation during the Data Transfer Phase. If "no use of Receipt Confirmation" is indicated or the NL entity cannot support the D-bit Procedure, then bit 7 is set to 0.

When an NL entity receives an INCOMING CALL packet with bit 7 of the GFI set to 1 but it cannot support the D-bit Procedure, it indicates "no use of Receipt Confirmation" in the Receipt Confirmation Selection parameter of the N-CONNECT indication primitive signaled to the Called NS user. Otherwise, if bit 7 of the GFI is set to 1 (respectively, 0), then the NL entity indicates "use (respectively, no use) of Receipt Confirmation" in the Receipt Confirmation Selection parameter of the N-CONNECT indication primitive signaled to the Called NS user.

When an NL entity receives an N-CONNECT response primitive with the Receipt Confirmation Selection parameter indicating "use (respectively, no use) of Receipt Confirmation", it sets bit 7 of the GFI in the CALL ACCEPTED packet to 1 (respectively, 0).

When an NL entity receives a CALL CONNECTED packet with bit 7 of the GFI set to 1 (respectively, 0), it indicates "use (respectively, no use) of Receipt Confirmation" in the Receipt Confirmation Selection parameter of the N-CONNECT confirm primitive signaled to the Calling NS user.

6.2.4 *Expedited Data Selection*

The Expedited Data Negotiation (EDN) Facility of the X.25/PLP is mapped to/from the Expedited Data Selection parameter of N-CONNECT primitives, when appropriate.

If the Expedited Data Selection parameter of the N-CONNECT request primitive indicates "use of Expedited Data", then the NL entity, if it can support the Interrupt Procedure using 32-octet INTERRUPT packets, encodes the EDN Facility to indicate use of expedited data during the Data Transfer Phase. If "no use of Expedited Data" is indicated or the NL entity cannot support 32-octet INTERRUPT packets, then the EDN Facility is omitted.

When an NL entity receives an INCOMING CALL packet with no EDN Facility or with the EDN Facility indicating use of expedited data but it cannot support 32-octet INTERRUPT packets, it indicates "no use of Expedited Data" in the Expedited Data Selection parameter of the N-CONNECT indication primitive signaled to the Called NS user. Otherwise, if the EDN Facility indicates use of expedited data, then the NL entity indicates "use of Expedited Data" in the Expedited Data Selection parameter of the N-CONNECT indication primitive signaled to the Called NS user.

When an NL entity receives an N-CONNECT response primitive with the Expedited Data Selection parameter indicating "use of Expedited Data", it encodes the EDN Facility in the CALL ACCEPTED packet to indicate use of expedited data. If the Expedited Data Selection parameter indicates "no use of Expedited Data", the NL entity omits the EDN Facility from the CALL ACCEPTED packet.

When an NL entity receives a CALL CONNECTED packet with the EDN Facility indicating use of expedited data, it indicates "use of Expedited Data" in the Expedited Data Selection parameter of the N-CONNECT confirm primitive signaled to the Calling NS user. If the CALL CONNECTED packet has no EDN Facility, then the NL entity indicates "no use of Expedited Data" to the Calling NS user.

6.2.5 *QOS Parameter Set*

The set of QOS parameters that are conveyed during the NC Establishment Phase consists of three , parameters:

a) the throughput for the direction of data transfer from the Calling NS user to the Called NS user;

b) the throughput for the direction of data transfer from the Called NS user to the Calling NS user; and

c) the transit delay that applies to both directions of data transfer.

Note – The mapping of the Protection and Priority QOS parameters in CCITT Recommendation X.213 to the corresponding DTE Facilities described in CCITT Recommendation X.25 is for further study.

For each of these three parameters, a set of "subparameters" is defined as follows:

a) a "Target" value, which is the QOS value desired by the Calling NS user;

b) a "Lowest Quality Acceptable" value, which is the lowest QOS value agreeable to the Calling NS user;

c) an "Available" value, which is the QOS value the NS provider is willing to provide; and

d) a "Selected" value, which is the QOS value to which the Called NS user agrees.

The set of values that can be specified for each subparameter is defined in every Network Service. This set includes the value "unspecified". It may also include a value defined to be a "default value" that is mutually understood by the NS provider and an NS user as applying in the absence of particular values.

6.2.5.1 *Throughput QOS Parameters*

The Throughput Class Negotiation (TCN) Facility and the Minimum Throughput Class Negotiation (MTCN) Facility of the X.25/PLP are mapped to/from both Throughput QOS parameters of N-CONNECT primitives, when appropriate. The specific mapping of the X.25/PLP facilities to/from both sets of Throughput subparameters is given in Table 3/X.223.

TABLE 3/X.223

Mapping of Throughput QOS Subparameters to X.25/PLP Facilities

CONS		X.25/PLP	
Subparameter	Primitive	Facility	Packet
Target	N-CONNECT request	TCN	CALL REQUEST
Lowest Quality Acceptable	N-CONNECT request	MTCN	CALL REQUEST
Available	N-CONNECT indication	TCN	INCOMING CALL
Lowest Quality Acceptable	N-CONNECT indication	MTCN	INCOMING CALL
Selected	N-CONNECT response	TCN	CALL ACCEPTED
Selected	N-CONNECT confirm	TCN	CALL CONNECTED

The set of values that can be specified for each Throughput subparameter ranges from 75 bits per second through 64000 bits per second, inclusive. This set consists of the following discrete values: 75, 150, 300, 600, 1200, 2400, 4800, 9600, 19200, 48000 and 64000 bits per second. An NL entity supports either all of these values or a contiguous subset of them. The value "unspecified" is also allowed.

6.2.5.1.1 *Processing an N-CONNECT Request Primitive*

If an NL entity, when receiving an N-CONNECT request primitive, cannot support the Lowest Quality Acceptable throughput (i.e., the minimum throughput) when specified for either direction of data transfer, then it rejects the request. In this case, the NL entity does not transmit any X.25/PLP packet but it does not signal an N-DISCONNECT indication primitive to the Calling NS user. The Originator parameter is "NS Provider". The Reason parameter is "Connection Rejection-QOS Not Available/Transient Condition", or "Connection Rejection-QOS not Available/Transit Condition", or "Connection Rejection-QOS Not Available/Permanent Condition" if the NL entity could never support the Lowest Quality Acceptable for either direction of data transfer.

If an NL entity, when receiving the N-CONNECT request primitive, can support the Lowest Quality Acceptable throughput (i.e., the minimum throughput) when specified for both directions of data transfer, then it encodes the Target and Lowest Quality Acceptable values in the TCN and MTCN Facilities, respectively (as shown in Table 3/X.223). If the Target subparameter (of either or both of the Throughput QOS parameters) is "unspecified", then the NL entity encodes the TCN Facility for the corresponding direction(s) of data transfer as the highest throughput rate supported by the NL entity. If the Lowest Quality Acceptable subparameter (of one of the Throughput QOS parameters) is "unspecified", then the NL entity encodes the MTCN Facility for the corresponding direction(s) of data transfer as 75 bits per second. The TCN and MTCN Facilities are transmitted across the DTE/DXE interface in a CALL REQUEST packet.

If the NL entity receives a N-CONNECT request primitive with the Lowest Quality Acceptable subparameters of both the Throughput QOS parameters "unspecified", then the MTCN facility is not included in the CALL REQUEST packet.

6.2.5.1.2 *Processing an INCOMING CALL Packet*

When receiving an INCOMING CALL packet, an NL entity compares the minimum throughput value specified in the MTCN Facility for each direction of data transfer to the available throughput value specified in the TCN Facility, when these are present. If, for either direction, the available throughput value is less than the minimum throughput value or if the NL entity cannot support the minimum throughput value, then the NL entity clears the call (i.e., transmits a CLEAR REQUEST packet). The cause is "DTE Originated" and the diagnostic is "Connection Rejection-QOS Not Available/Transient Condition", or "Connection Rejection-QOS Not Available/ Permanent Condition" if the NL entity could never support the lowest throughput value (these diagnostics have values 229 and 230, respectively). Otherwise, the NL entity indicates, for both directions of data transfer, the Available and Lowest Quality Acceptable throughput values in the Throughput QOS parameters of the N-CONNECT indication primitive signalled to the Called NS user. The Available and Lowest Quality Acceptable subparameters are mapped from the TCN and MTCN Facilities, respectively, as shown in Table 3/X.223.

If an NL entity receives an INCOMING CALL packet without the MTCN Facility, then the NL entity indicates the value "unspecified" for the Lowest Quality Acceptable sub-parameters of both the Throughput QOS parameters of the N-CONNECT indication primitive signalled to the called NS user.

6.2.5.1.3 *Processing an N-CONNECT Response Primitive*

When an NL entity receives an N-CONNECT response primitive, it encodes the Selected throughput values for both directions of data transfer, as given in the Throughput QOS parameters, in the TCN Facility returned in the CALL ACCEPTED packet.

6.2.5.1.4 *Processing a CALL CONNECTED packet*

When an NL entity receives a CALL CONNECTED packet, it indicates the Selected throughput values for both directions of data transfer, as given in the TCN Facility, in the Throughput QOS parameters of the N-CONNECT confirm primitive signalled to the Calling NS user.

6.2.5.2 *Transit Delay QOS Parameter*

The Transit Delay Selection And Indication (TDSAI) Facility and the End-to-End Transit Delay Negotiation (EETDN) Facility of the X.25/PLP are mapped to/from the Transit Delay QOS parameter of N-CONNECT primitives, when appropriate.

The set of values that can be specified for each Transit Delay subparameter ranges from 1 millisecond through 65534 milliseconds, inclusive, in increments of 1 millisecond. An NL entity supports either all of these values or a contiguous subset of them. The value "unspecified" is also allowed.

An NL entity in an end system shall be able to determine the cumulative transit delay attributable to the NS provider in that end system. This is the transit delay of the NL entity itself, all lower-layer entities, and the effects of the access line transmission rate.

Appendix III illustrates the use of the X.25 TDSAI and EETDN Facilities in support of the end-to-end negotiation of the Transit Delay QOS parameter.

6.2.5.2.1 *Processing an N-CONNECT Request Primitive*

If an NL entity, when receiving an N-CONNECT request primitive, cannot support the Lowest Quality Acceptable transit delay (i.e., the maximum transit delay), when specified, then it rejects the request. In this case, the NL entity does not transmit any X.25/PLP packet but it does signal an N-DISCONNECT indication primitive to the Calling NS user. The originator parameter is "NS Provider". The Reason parameter is "Connection Rejection-QOS Not Available/Transient Condition", or "Connection Rejection-QOS Not Available/Permanent Condition" if the NL entity could never support the Lowest Quality Acceptable transit delay.

If an NL entity, when receiving an N-CONNECT request primitive, can support the Lowest Quality Acceptable transit delay (i.e., the maximum transit delay) when specified, or when the Target transit delay is specified and the Lowest Quality Acceptable delay is unspecified, then:

a) the NL entity encodes the cumulative transit delay attributable to the NS provider in the calling end system in the "cumulative-transit-delay subfield" (i.e., octets 1 and 2) of the EETDN Facility;

b) if a Target transit delay is specified, then the NL entity encodes this value in the "target-transit-delay subfield" (i.e., octets 3 and 4) of the EETDN Facility (otherwise, this subfield is not used);

   Note – According to Recommendation X.213, the case where the Target transit delay is unspecified and the Lowest Quality Acceptable transit delay has a value other than unspecified is not permitted; logically, this case can be represented by the permitted assignment where an identical value is specified for both the Target and Lowest Quality Acceptable transit delays;

c) if a Lowest Quality Acceptable transit delay is specified, then the NL entity encodes this value in the "maximum-acceptable-transit-delay subfield" (i.e., octets 5 and 6) of the EETDN Facility (otherwise, this subfield is not used); and

d) if the Target transit delay is specified and the operational environment is DTE-to-DCE, then the NL entity encodes the value of the TDSAI Facility as being less than the Target transit delay minus the cumulative transit delay for the calling end system; otherwise, the TDSAI Facility is encoded with any value (i.e., it is not constrained by this Recommendation). In a DTE-to-DTE operational environment, the usage of the TDSAI Facility is for further study.

   Note – Given a "routing management information base", the NL entity can refine the value encoded in the TDSAI Facility. For example, the value of the TDSAI Facility could take into account whether networks other than packet-switched networks are traversed in reaching the called end system or whether the called end system is reachable directly in a point-to-point configuration.

The TDSAI (if applicable) and EETDN Facilities are transmitted across the DTE/DXE interface in a CALL REQUEST packet.

Note – The value of the TDSAI Facility in a CALL REQUEST packet in a DTE/DCE environment provides a guideline to the DCE for allocating resources. The final transit-delay value applicable to the Virtual Call may be less than, equal to, or greater than the value in the CALL REQUEST packet.

If an NL entity receives an N-CONNECT request primitive with both Target and Lowest Quality Acceptable sub-parameters "unspecified", then the EETDN and the TDSAI (if applicable) is not included in the CALL REQUEST packet.

6.2.5.2.2 *Processing an INCOMING CALL packet*

When receiving an INCOMING CALL packet, an NL entity computes the total NC transit delay by summing the values of:

a) the TDSAI Facility;

b) the "cumulative-transit-dekay subfield" (i.e., octets 1 and 2) of the EETDN Facility; and

c) the transit delay attributable to the NS provider in the called end system.

Note – The procedure suggested here for computing the value of the total NC transit delay is the best an NL entity can do in the absence of any "external information". However, given a "routing management information base", the NL entity can refine this value. For example, the transit delay attributable to the effects of the access line transmission rate shall not be included when the called end system is connected to the calling end system in a point-to-point configuration (these effects have been accounted for by the calling end system).

If the "maximum-acceptable-transit-delay subfield" (i.e., octets 5 and 6) of the EETDN Facility is present, then the NL entity compares the value in this "subfield" to the total NC transit delay computed above. If the total NC transit delay is greater than the maximum-acceptable transit delay, then the NL entity clears the call (i.e., transmits a CLEAR REQUEST packet). The cause is "DTE Originated" and the diagnostic is "Connection Rejection-QOS Not Available/Transient Condition", or "Connection Rejection-QOS Not Available/Permanent Condition" if the NL entity could never support the maximum-acceptable transit delay (these diagnostics have values 229 and 230, respectively). Otherwise, if either:

1) the total NC transit delay is less than or equal to the maximum-acceptable transit delay, or

2) the "maximum-acceptable-transit-delay subfield" of the EETDN Facility is not present,

then the NL entity indicates the Available transit-delay value (as given by the total NC transit delay computed above) in the Transit Delay QOS parameter of the N-CONNECT indication primitive signaled to the Called NS user.

If an NL entity receives an INCOMING CALL packet without the EETDN or without the TDSAI Facilities, then the NL entity indicates the value "unspecified" for the Available transit delay of the N-CONNECT indication primitive signalled to the called NS user.

6.2.5.2.3 *Processing an N-CONNECT Response Primitive*

When an NL entity receives an N-CONNECT response primitive, it encodes the total NC transit-delay value (as computed above) in the "cumulative-transit-delay subfield" (octets 1 and 3) of the EETDN Facility returned in the CALL ACCEPTED packet.

Note 1 – There is no Transit Delay QOS Parameter in an N-CONNECT response primitive.

Note 2 – The EETDN Facility returned in a CALL ACCEPTED packet only contains the "cumulative-transit-delay subfield".

If a N-CONNECT response primitive is received by an NL entity subsequent to a N-CONNECT indication primitive signalled from the NL entity with the Available transit delay parameter set to "unspecified", then the NL entity does not include the EETDN in the CALL ACCEPTED packet.

6.2.5.2.4 *Processing a CALL CONNECTED Packet*

When an NL entity receives a CALL CONNECTED packet, it indicates the selected transit-delay value, as given by the "cumulative-transit-delay subfield" of the EETDN Facility, in the Transit Delay QOS parameter of the N-CONNECT confirm primitive signaled to the Calling NS user.

If a NL entity receives a CALL CONNECTED packet without the EETDN Facility, then the NL entity indicates an "unspecified" value in the N-CONNECT confirm primitive signalled to the NS user.

6.2.6 *NS-User-Data*

The Call User Data Field of X.25/PLP CALL REQUEST and INCOMING CALL packets is used to transer the NS-user-data of N-CONNECT request and indication primitives, respectively. The Called User Data Field of X.25/PLP CALL ACCEPTED and CALL CONNECTED packets is used to transfer the NS-user-data of N-CONNECT response and confirm primitives, respectively. In addition, the Fast Select Facility shall be indicated in the CALL REQUEST packet sent by the Calling NL entity.

7 Network Connection Release Phase

7.1 *Primitive/Parameter and Packet/Field Relationships*

Table 4/X.223 shows the relationships between the primitives/parameters used during the NC Release Phase and the packets/fields associated with the Call Clearing Procedures.

TABLE 4/X.223

CONS:X.25/PLP Mapping for the Network Connection Release Phase

CONS	X.25/PLP
PRIMITIVES:	PACKETS:
N-DISCONNECT request	CLEAR REQUEST
N-DISCONNECT indication	CLEAR INDICATION, RESTART INDICATION [1], CLEAR REQUEST [2]
PARAMETERS:	FIELDS (INCLUDING FACILITIES):
Originator and Reason	Cause Code and Diagnostic Code Fields [3]
NS-User-Data	Clear User Data
Responding Address	Called DTE Address Field Called Address Extension Facility

Note 1 – Receipt of a RESTART INDICATION packet should be treated as receipt of a CLEAR INDICATION packet for every logical channel and then mapped to an N-DISCONNECT indication primitive for every active NC associated with the Packet Layer Protocol being restarted. The Restarting Cause Code and Diagnostic Code Fields are then treated in the same manner as the Clearing Cause Code and Diagnostic Code Fields.

Note 2 – See § 7.2.1, Paragraph 2.

Note 3 – The combination of Cause Code and Diagnostic Code Fields is mapped to/from the combination of Originator and Reason parameters.

7.2 *Procedures*

7.2.1 *Primitive/Packet Mapping*

When an NL entity receives an N-DISCONNECT request primitive from an NS user, it transmits a CLEAR REQUEST packet across the DTE/DXE interface. If, however, the NL entity had previously transmitted a CLEAR REQUEST packet and signaled an N-DISCONNECT indication primitive to the NS user (because of a protocol error; see below), then it does not transmit another CLEAR REQUEST packet.

If an NL entity detects an error in the operation of the X.25/PLP for which its action is to clear the VC (e.g., a format error in an INCOMING CALL packet or a timeout condition), then it transmits a CLEAR REQUEST packet across the DTE/DXE interface. If the virtual circuit is associated with an NC, then it also signals an N-DISCONNECT indication primitive to the NS user.

When an NL entity receives a CLEAR INDICATION packet (or a RESTART INDICATION packet), it signals an N-DISCONNECT indication primitive to the NS user. It also transmits a CLEAR CONFIRMATION packet (or a RESTART CONFIRMATION packet) across the DTE/DXE interface. If, however, the NL entity had previously transmitted a CLEAR REQUEST packet for the NC (i.e., a clear collision), then it does not signal an N-DISCONNECT indication primitive to the NS user nor transmit a CLEAR CONFIRMATION packet.

Note – If the received CLEAR INDICATION packet is in response to a previously-transmitted CALL REQUEST packet, the NL entity may retry the call if the Network Connection Establishment Delay has not been exceeded rather than immediately signalling an N-DISCONNECT indication primitive to its NS user. The NL entity may also use the Clearing Cause Code (see § 7.2.2) in the CLEAR INDICATION packet to determine whether to retry the call. That is, the reattempt may be successful if the Clearing Cause Code is classified as Category C (see Recommendation X.96); on the other hand, a Category D code indicates a problem of a more permanent nature. The time interval between and number of reattempted calls is a local matter. If multiple attempts at establishing the NC are all unsuccessful, then the Originator-parameter and Reason-parameter values finally signaled in the N-DISCONNECT indication primitive are a local matter.

If either NL entity wishes to disconnect an NC, it signals an N-DISCONNECT indiction primitive to its NS user and transmits a CLEAR REQUEST packet across the DTE/DXE interface. If, however, the NL entity in the calling DTE cannot, for example, support the QOS parameters specified in an N-CONNECT request primitive or does not have an LC available to set up a VC, then it signals an N-DISCONNECT indication primitive to the Calling NS user but it does not transmit a CLEAR REQUEST packet across the DTE/DXE interface.

7.2.2 *Originator/Reason*

The combination of Originator and Reason parameters of the N-DISCONECT primitives is mapped to/from the combination of Clearing Cause Code (or Restarting Cause Code) and the Diagnostic Code Fields.

The combination of the cause code "DTE Originated" (coded as all zeros) with a diagnostic in the set 241, 242 and 244-248 corresponds to an Originator-parameter values of "NS User". In this case, there is a one-to-one relationship between the values of the Reason parameter and these diagnostic codes.

The cause code "DTE Originated" (coded as all zeros) used in combination with diagnostic codes other than those listed above corresponds to an Originator-parameter value of "NS Provider". There is a one-to-one relationship between the values of the Reason parameter and diagnostic codes 225-232 and 235.

In other cases, the Originator-parameter and Reason-parameter values depend on:

a) the cause and/or diagnostic codes; and

b) whether the NC is in the NC Establishment Phase or in the Data Transfer Phase.

The values of the Originator and Reason parameters are derived as follows:

a) the Originator-parameter value is "NS Provider" and the Reason-parameter value is "disconnection-permant condition" when the NC is in the Data Transfer Phase and any of the following applies:

   – cause codes "Out of Order", "Local Procedure Error", "Remote Procedure Error", or "RPOA Out of Order";

   – diagnostic code 122;

b) the Originator-parameter value is "NS Provider" and the Reason-parameter value is "disconnection-transient condition" when the NC is in the Data Transfer Phase and any of the following applies:

   – cause code "Network Congestion";

   – diagnostic codes 113 or 115;

   – cause code "DTE Originated" (coded as all zeros) with diagnostic codes 162 or 163;

c) the Originator-parameter value is "NS Provider" and the Reason-parameter value is "connection rejection-NSAP address unknown (permanent condition)" when the NC is in the NC Establishment Phase and any of the following applies:

   – cause codes "Not Obtainable" or "Ship Absent";

d) the Originator-parameter value is "NS Provider" and the Reason-parameter value is "connection rejection-reason unspecified/permanent condition" when the NC is in the NC Establishment Phase and any of the following applies:

- cause codes "Access Barred", "Fast Select Acceptance Not Subscribed", "Incompatible Destination", "Invalid Facility Request", "Out of Order", "Local Procedure Error", "Remote Procedure Error", "Reverse Charging Acceptance Not Subscribed", or "RPOA Out of Order";
- diagnostic codes 121 or 122;
- cause code "DTE Originated" (coded as all zeros) with diagnostic code 164;

e) the Originator-parameter value is "NS Provider" and the Reason-parameter value is "connection rejection-reason unspecified/transient condition" when the NC is in the NC Establishment Phase and any of the following applies:

- cause codes "Network Congestion" or "Number Busy";
- diagnostic codes 112-120;
- cause code "DTE Originated" (coded as all zeros) with a diagnostic code other than those listed above;

f) the Originator-parameter and Reason-parameter values are both "Undefined" for any other combination of cause and diagnostic codes.

7.2.3 *NS-User Data*

The Clear User Data Field of X.25/PLP CLEAR REQUEST and CLEAR INDICATION packets is used to transfer the NS-user-data between NS users.

7.2.4 *Responding Address*

Local operation determines the contents of the Called Address Field and whether the responding NSAP Address, where explicitly supplied, is mapped to/from the AF or the AEF in the X.25/PLP call clearing packets. Rules for encoding and decoding the responding NSAP Address are given in Clause § 6.2.2.

8 Data Transfer Phase – Data Transfer Service

8.1 *Primitive/Parameter and Packet/Field Relationships*

Table 5/X.223 shows the relationships between the primitives/parameters used for the Data Transfer Service and the packets/fields associated with the Data Transfer Procedures.

TABLE 5/X.223

CONS:X.25/PLP Mapping for the Data Transfer Service

CONS	X.25/PLP
PRIMITIVES:	PACKETS:
N-DATA request	DATA
N-DATA indication	DATA
PARAMETERS:	FIELDS:
NS-User-Data	User Data, M-bit
Confirmation Request	D-bit, P(S)

8.2 *Procedures*

8.2.1 *Primitive/Packet Mapping*

When an NL entity receives an N-DATA request primitive from an NS user, it transmits a sequence of one or more DATA packets, known as M-bit Sequence (MBS), across the DTE/DXE interface. The number of DATA packets needed in an MBS depends on the amount of NS-user-data and on the maximum "packet size" (i.e., the maximum User Data Field Length of DATA packets) permitted on the DTE/DXE interface. All DATA packets but the last one of an MBS contain the maximum number of octets, have their M-bit set to 1, and have their D-bit set to 0. The last DATA packet has its M-bit set to 0. The D-bit setting of the last DATA packet is dependent on the Confirmation Request parameter (see § 8.2.3 below).

When an NL entity receives an MBS, it signals an N-DATA indication primitive to the NS user.

8.2.2 *NS-User-Data*

The User Data Fields of X.25/PLP DATA packets are used to transfer NS-user-data between NS users.

8.2.3 *Confirmation Request*

The D-bit of the last DATA packet in an MBS is mapped to/from the Confirmation Request parameter.

If an N-DATA request primitive indicates in the Confirmation Request parameter that confirmation of receipt is requested (respectively, not requested), then the D-bit of the last DATA packet in an MBS is set to 1 (respectively, 0). In the case of confirmation of receipt being requested, the NL entity shall use a locally-defined mechanism to associate the P(S) of the last DATA packet in the MBS with the N-DATA request primitive. (This mechanism shall also provide for an association of an N-DATA request primitive with an N-DATA ACKNOWLEDGE indication primitive; see § 9.2.1).

When an NL entity signals an N-DATA indication primitive to the NS user, it indicates in the Confirmation Request parameter that confirmation of receipt is requested (respectively, not requested) if the D-bit of the last DATA packet in an MBS is set to 1 (respectively, 0). When the last DATA packet in an MBS has its D-bit set to 1, the NL entity may not transmit a P(R) corresponding to that DATA packet across the DTE/DXE interface until it receives an N-DATA ACKNOWLEDGE request primitive from its NS user (see § 9). In the case of the D-bit of the last DATA packet in an MBS being set to 1, the NL entity shall use a locally-defined mechanism to associate the P(S) of this packet with the N-DATA indication primitive. (This mechanism shall also provide for an association of an N-DATA indication primitive with an N-DATA ACKNOWLEDGE request primitive; see § 9.2.1.)

9 **Data Transfer Phase – Receipt Confirmation Service**

9.1 *Primitive and Packet/Field Relationships*

There is no distinct X.25/PLP packet associated with the N-DATA ACKNOWLEDGE request and N-DATA ACKNOWLEDGE indication primitives. The P(R) field of DATA, RECEIVE READY, RECEIVE NOT READY, and REJECT (if agreed to) packets is used to support the Receipt Confirmation Service.

9.2 *Procedures*

9.2.1 *Primitive/Packet Mapping*

When an NL entity receives an N-DATA ACKNOWLEDGE request primitive from an NS user, it uses its locally-defined mechanism mentioned in § 8.2.3 for associating an N-DATA ACKNOWLEDGE request primitive with a previously-issued N-DATA indication primitive [and, hence, a P(S)] to determine a P(R) to be transferred in the appropriate packet across the DTE/DXE interface. (Note that such acknowledgements shall be issued in the same order that the corresponding N-DATA indications were issued.)

When an NL entity receives a P(R), it shall determine whether this P(R) is inclusive of a P(S) associated with a previously-received N-DATA request primitive that requested confirmation of receipt. If such an association is made, then the NL entity signals an N-DATA ACKNOWLEDGE indication primitive to the NS user. This N-DATA ACKNOWLEDGE indication primitive is associated, by the locally-defined mechanism mentioned in § 8.2.3, to the previously-received N-DATA request primitive that had requested confirmation of receipt.

10 Data Transfer Phase – Expedited Data Transfer Service

10.1 *Primitive/Parameter and Packet/Field Relationships*

Table 6/X.223 shows the relationships between the primitives/parameters used for the Expedited Data Transfer Service and the packets/fields associated with the Interrupt Transfer Procedures.

TABLE 6/X.223

CONS:X.25/PLP Mapping for the Expedited Data Transfer Service

CONS	X.25/PLP
PRIMITIVES:	PACKETS:
N-EXPEDITED DATA request	INTERRUPT
N-EXPEDITED DATA indication	INTERRUPT
PARAMETERS:	FIELDS:
NS-User-Data	Interrupt User Data

10.2 *Procedures*

10.2.1 *Primitive/Packet Mapping*

When an NL entity receives an N-EXPEDITED DATA request primitive from an NS user, it transmits an INTERRUPT packet across the DTE/DXE interface. An NL entity shall not transmit a second INTERRUPT packet before an outstanding INTERRUPT packet has been confirmed by an INTERRUPT CONFIRMATION packet.

When an NL entity receives an INTERRUPT packet, it signals an N-EXPEDITED DATA indication primitive to the NS user. It also transmits an INTERRUPT CONFIRMATION packet across the DTE/DXE interface.

10.2.2 *NS-User-Data*

The Interrupt User Data Field of X.25/PLP INTERRUPT packets is used to transfer expedited NS-user-data between NS users.

11 Data Transfer Phase – Reset Service

11.1 *Primitive/Parameter and Packet/Field Relationships*

Table 7/X.223 shows the relationships between the primitives/parameters used for the Reset Service and the packets/fields associated with the Reset Procedures.

TABLE 7/X.223

CONS:X.25/PLP Mapping for the Reset Service

CONS	X.25/PLP
PRIMITIVES:	PACKETS:
N-RESET request	RESET REQUEST
N-RESET indication	RESET INDICATION, RESET REQUEST [1]
N-RESET response	none
N-RESET confirm	none
PARAMETERS:	FIELDS:
Originator and Reason	Cause Code and Diagnostic Code Fields [2]

Note 1 – See Clause 11.2.1, Paragraph 2.

Note 2 – The combination of Cause Code and Diagnostic Code Fields is mapped to/from the combination of Originator and Reason parameters.

11.2 *Procedures*

11.2.1 *Primitive/Packet Mapping*

When an NL entity receives an N-RESET request primitive from an NS user, it transmits a RESET REQUEST packet across the DTE/DXE interface. When the NL entity is ready to accept subsequent data, expedited data, and confirmations of receipt from the NS user, it signals an N-RESET confirm primitive. The issuing of this primitive may or may not be related to the completion of the X.25/PLP reset procedure. Any data or expedited data received from the NS user following the N-RESET confirm primitive is transmitted after completion of the X.25/PLP reset procedure.

If an NL entity detects an error in the operation of the X.25/PLP for which its action is to reset the virtual circuit (e.g., a sequence error or a timeout condition), then it transmits a RESET REQUEST packet across the DTE/DXE interface. When an NL entity is ready to accept subsequent data, expedited data, and confirmations of receipt from the NS user, it signals an N-RESET indication primitive. The issuing of this primitive may or may not be related to the completion of the X.25/PLP reset procedure. Any data or expedited data received from the NS user following the N-RESET response primitive is transmitted after completion of the X.25/PLP reset procedure.

When an NL entity receives a RESET INDICATION packet, it signals an N-RESET indication primitive to the NS user.

When an N-RESET response primitive is received from the NS user, the NL entity shall be willing to accept the subsequent data, expedited data, and confirmations of receipt received from the NS user for transmission upon completion of the X.25/PLP reset procedure.

During the reset process, the following actions are taken by the NL entity with respect to the operation of the X.25/PLP:

a) For DATA packets:

   - those awaiting transmission may either be transmitted prior to transmitting a reset packet or flushed from the queue of DATA packets awaiting transmission;

   - those remaining in the transmit window when the reset procedure is completed are flushed; and

   - those that have been received prior to receiving a reset packet but which do not constitute an entire MBS are flushed from the "MBS reassembly area".

b) The lower window edge for each direction of data transmission is set to 0 and subsequently transmitted DATA packets are numbered starting from 0.

c) Any busy condition that had existed prior to the reset is considered not to exist any longer.

d) Any outstanding INTERRUPT packet remains unconfirmed.

e) All timer and retransmission parameters relating to data and interrupt transfer are set back to their initial value.

No action is required with respect to the provision of the Network Service by an NL entity when it receives a RESET CONFIRMATION packet or a RESET INDICATION packet in response to a RESET REQUEST packet (i.e., a reset collision). However, it shall then be capable of receiving subsequent DATA and INTERRUPT packets and P(R) information.

11.2.2 *Originator/Reason*

The combination of Originator and Reason parameters of the N-RESET primitives is mapped to/from the combination of Resetting Cause Code and Diagnostic Code Fields.

The combination of the cause code "DTE Originated" (coded as all zeros) with the diagnostic "Reset-User Resynchronization" (diagnostic code 250) corresponds to an Originator-parameter value of "NS User" and a Reason-parameter value identical to the diagnostic.

All other combinations of cause codes, except "DTE Originated" coded as "10000000", and diagnostic codes specified in Recommendation X.25, corresponds to an Originator-parameter value of "NS Provider". The value of the Reason parameter is derived as follows:

a) "congestion" if any of the following applies:

   - cause code "Network Congestion";

   - cause code "DTE Originated" (coded as all zeros) and diagnostic 234;

b) "reason unspecified" for any other combination of cause and diagnostic codes.

The cause code "DTE Originated" coded as "10000000" with any diagnostic code, as well as cause codes not specified in Recommendation X.25 with any diagnostic code, corresponds to values of both the Originator parameter and the Reason parameter of "Undefined".

APPENDIX I

(to Recommendation X.223)

Additional Considerations of CONS Primitives

I.1 *Introduction*

The main body of this Recommendation presents a mapping between the Connection-Mode Network Service (CONS) primitives and the X.25/Packet Layer Protocol (PLP) elements. However, the designer of an end system should be aware that there are several issues related to the issuing of CONS primitives in addition to mapping them to X.25/PLP protocol elements. These issues realte to the provision of the appropriate "environment" (i.e. supporting protocols at apropriate layers) within the end system in which the X.25/PLP is to operate. The purpose of this appendix is to briefly describe these issues.

I.2 *Environment for X.25/PLP operation*

For the purpose of this appendix, the environment in which the X.25/PLP operates depends on the technology of the subnetwork(s) to which the end system is attached. For example, the end system may be attached to a packet-switched public data network. While the mapping between the primitives of the CONS and the elements of the X.25/PLP does not depend on the particular subnetwork, the proprer provision of the environment for the X.25/PLP to operate does depend on it. The following subsections address the issues pertaining to provision of the environment in which the X.25/PLP operates.

I.2.1 *Initialization*

If, when receiving an N-CONNECT request primitive, the Network Layer (NL) entity determines that the necessary Subnetwork Point of Attachment (SNPA) in this end system is not available (i.e. cannot be used for transmitting CALL REQUEST packet), then appropriate procedures are necesary to be executed in the end system to make the SNPA available. Alternatively, the NL entity may reject the request. In this case, the corresponding procedures are not executed and the NL entity signals an N-DISCONNECT indication primitive to the Calling Network Service (NS) user. The Originator parameter is "NS Provider" and the Reason parameter is "Connection Rejection-Reason Unspecified/Permanent Condition."

Note– It is beyond the scope of this Recommendation to indicate how the NL entity determines whether the necessary SNPA is or is not available.

It is not the intent of this appendix to provide a complete set of procedures that are executed for the various subnetwork technologies in which the X.25/PLP may be used. Still, an example will provide an indication of these procedures.

Example: *Connection of an End System to an X.25 Packet-Switched Data Network*

Consider an end system connected to an X.25 packet-switched public data network by a dedicated line conforming to Recommendation X.21. If this interface is not available when an N-CONNECT request primitive is received by the NL entity, then the following steps are taken (in the order shown):

a) the X.21 establishment procedures are performed and the X.21 data transfer phase is entered;

b) the LAPB procedures are executed to establish the Link Level of the X.25 DTE/DCE interface and enter its data transfer phase; and

c) the X.25/PLP restart procedure is executed.

Only after the successful completion of all three steps above can the NL entity transfer an X.25/PLP CALL REQUEST packet across the DTE/DCE interface.

It is also not the intent of this appendix to indicate how the NL entity is informed of the outcome of the initialization procedures. However, it is asumed that the NL entity is informed whether these procedures are successfully completed. The subsequent action of the NL entity depends on the outcome for example:

a) Successful Initialization: the NL entity transmits a CALL REQUEST packet; or

b) Unsuccessful Initialization: the NL entity may reattempt the initialization procedures again or signal an N-DISCONNECT indication primitive to the NS user but without transmitting a CLEAR REQUEST packet. In the latter case, the Originator parameter is "NS Provider." The Reason parameter is "Connection Rejection-Reason Unspecified/Transient Condition".

Note – A more detailed mapping of the Reason parameter to any diagnostic information available as a result of the failure of the initialization procedures may also be desired.

In a similar fashion as above for an N-CONNECT request primitive, it should be recognized that the initialization procedures must be completed before an N-CONNECT indication primitive can be signaled to an NS user.

1.2.2 *Premature Closedown*

If the environment in which the X.25/PLP operates prematurely closes down (i.e. while one or more NCs are established or in the process of being established), then the NL entity signals, for each established NC or NC in the process being estblished, an N-DISCONNECT indication primitive to the NS user but does not transmit a CLEAR REQUEST packet. The Originator parameter is "NS Provider." The Reason parameter is:

a) for established NCs, "Disconnection-Transient Condition," or

b) for NCs in the process of being established, "Connection Rejection-Transient Condition".

Note – A more detailed mapping of the reason parameter to any diagnostic information available as a result of the premature closedown may also be desired.

APPENDIX II

(to Recommendation X.223)

Use of X.25/PLP NPAI

II.1 *Introduction*

This appendix discusses the use of X.25/PLP Network Protocol Address Information (NPAI), i.e., the Address Field (AF) and the Address Extension Facility (AEF). It provides guidelines for obtaining the Subnetwork Point of Attachment (SNPA) Address from the Network Services Acces Point (NSAP)Address. It also illustrates how an NSAP Address may be encoded in X.25/PLP NPAI.

II.2 *Obtaining an SNPA address*

Two methods for obtaining an SNPA Address from an NSAP Address are described. The first one makes use of a directory, the second describes an algorithmic procedure. The two methods are not exclusive.

II.2.1 *Directory*

The directory is an abstract object which, given an NSAP Address, returns an SNPA Address. The operation of such a directory is not within the scope of this appendix. Conceptually, it may be viewed as a table look-up, a local directory, or a distributed directory.

II.2.2 *Algorithmic Procedure*

There are three cases that may be considered for deriving an SNPA Address from an NSAP Address:

a) *Domain Specific Part (DSP) absent:*

1) The NSAP Address is composed of an Address and Format Identifier (AFI) and an Initial Domain Identifier (IDI). If the AFI is consistent with the AFI format of the subnetwork provider, the IDI may be used directly in the AF subject to network-dependent prefixes and formats to provide the encoded SNPA Address. In this case, the AFI is not conveyed as explicit protocol control information. Its existence is thus implied and must be capable of being correctly deduced by the recipient.

2) In the case where the AFI format of the NSAP Address is not consistent with the subnetwork provider, it may be necessary to make use of a directory as described in § II.1.1 above.

b) *DSP present:*

The procedure to be followed in this case requires that the IDI and AFI be operated on as specified in Case (a) above to determine the SNPA Address. The only difference for this case is that, in addition to the above, the complete NSAP Address is inserted in the AEF.

c) There may be cases, such as the use of escape digits (e.g. 8 = F.69, 9 = E.163), that do not require the use of directories. In cases such as this, the procedure defined in the appropriate addressing standard/recommendation (e.g. Recommendation X.121) may also be implied.

II.3 *Examples of NSAP Address Encoding*

Below are several examples of how an NSAP Address is encoded in X.25/PLP NPAI (i.e. the AF and the AEF). Section 6.2.2 specifies how this encoding is performed. As indicated, the preferred binary encoding, as defined in Recommendation X.213, is used as the encoding technique.

The examples make use of hexadecimal notation; that is X'h_1h_2...' represents a string of hexadecimal digits. Padding digits are highlighted with an underscore.

Example 1:

AFI	IDI	DSP
X'36'	X'313412345678'	null

Assuming the conditions in § 6.2.2.1.1 are all satisfied, the above NSAP Address is conveyed in the AF. The AF would then be encoded as:

AF
X'313412345678'

Note that the need to include the Data Network Identification Code, which is 3134 in this example, and any prefix digits is a matter dependent on the packet-switched network to which the end system is attached.

Example 2:

AFI	IDI	DSP
X'37'	X'31341234567890'	X'5F4230A26789'

This NSAP Address can only be conveyed in the AEF. The encoding of the Facility Parameter Field (FPF) of the AEF is as follows:

FPF of AEF

X'1C'	X'37313412345678905F4230A26789'

Note that the first octet of the FPF of the AEF indicates the usage of the AEF (in this case, full NSAP Addres) in bits 8 and 7 and the number of semi-octets that follow (twenty-eight) in bits 6, 5, 4, 3, 2 and 1.

Example 3:

AFI	IDI	DSP
X'44'	X'123456789012345'	X'4297'

This NSAP Address can only be conveyed in the AEF. The encoding of the FPF of the AEF is as follows:

FPF de l'AEF

X'16'	X'441234567890123454297F'

Example 4:

AFI	IDI	DSP
X'45'	X'1234567890123'	X'FE496A'

This NSAP Address can only be conveyed in the AEF. The encoding of the FPF of the AEF is as follows:

FPF of AEF

X'18'	X'45001234567890123FFE496A'

Example 5:

AFI	IDI	DSP
X'47'	X'4368'	X'43678A4B095ECF'

This NSAP Address can only be conveyed in the AEF. The encoding of the FPF of the AEF is as follows:

FPF of AEF

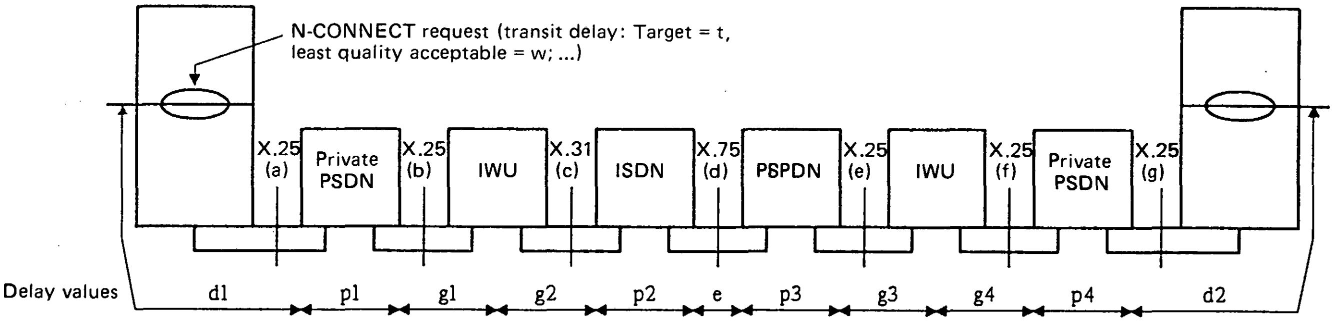
X'14'	X'47436843678A4B095ECF'

APPENDIX III

(to Recommendation X.223)

Transit Delay Calculations

This appendix illustrates how the various X.25 facilities are used to negotiate the end-to-end value of the transit delay QOS parameter.



T0702730-87

ISDN	Integrated Services Digital Network
IWU	Interworking Unit
PSDN	Packet Switched Data Network
PSPDN	Packet Switched Public Data Network

The labels (a), (b), (c), (d), (e), (f) and (g) represent the various points between the entities involved in the scenario shown above at which the transit delay information is visible in the protocol control information.

	X.25 Facility	X.75 Utilities		EETDN Facility		
	TDSAI	TDS	TDI	CTD	TTD	MATD
Call Request Phase						
a)	t − 2d1 (Note 1)	NA	NA	2d1	t	w
b)	p1	NA	NA	2d1	t	w
c)	t − 2d1 − p1 − (g1 + g2)	NA	NA	2d1 + p1 + (g1 + g2)	t	w
d)	NA	t − 2d1 − p1 − (g1 + g2)	p2 + e	2d1 + p1 + (g1 + g2)	t	w
e)	p2 + e + p3	NA	NA	2d1 + p1 + (g1 + g2)	t	w
f)	t − [2d1 + p1 + (g1 + g2)] − (g3 + g4) − (p2 + e + p3)	NA	NA	2d1 + p1 + (g1 + g2) + (p2 + e + p3) + (g3 + g4)	t	w
g)	p4	NA	NA	2d1 + p1 + (g1 + g2) + (p2 + e + p3) + (g3 + g4)	t	w
Call Confirmation Phase (Note 2)						
g)	NA	NA	NA	2d1 + p1 + (g1 + g2) + (p2 + e + p3) + (g3 + g4) + p4	NA	NA
f)	p4	NA	NA	2d1 + p1 + (g1 + g2) + (p2 + e + p3) + (g3 + g4) + p4	NA	NA
e)	NA	NA	NA	2d1 + p1 + (g1 + g2) + (p2 + e + p3) + (g3 + g4) + p4	NA	NA
d)	NA	NA	p2 + e + p3	2d1 + p1 + (g1 + g2) + (p2 + e + p3) + (g3 + g4) + p4	NA	NA
c)	p2 + e + p3	NA	NA	2d1 + p1 + (g1 + g2) + (p2 + e + p3) + (g3 + g4) + p4	NA	NA
b)	NA	NA	NA	2d1 + p1 + (g1 + g2) + (p2 + e + p3) + (g3 + g4) + p4	NA	NA
a)	p1	NA	NA	2d1 + p1 + (g1 + g2) + (p2 + e + p3) + (g3 + g4) + p4	NA	NA

Note 1 – The calling DTE asumes d2 is the same as d1.

Note 2 – The called DTE accepts the call if:

$$2d1 + p1 + (g1 + g2) + (p2 + e + p3) + (g3 + g4) + p4 \leqslant w.$$

CTD	Cumullative Transit Delay.
EETDN	End-to-End Transit Delay Negotiation (Facility).
MATD	Maximum-Acceptable Transit Delay.
NA	Not Applicable.
TDI	Transit Delay Indication (Utility).
TDS	Transit Delay Selection (Utility).
TDSAI	Transit Delay Selection and Indication (Facility).
TTD	Target Transit Delay.

APPENDIX IV

(to Recommendation X.223)

Differences between Recommendation X.223 and ISO 8878

Recommendation X.223 is technically aligned with ISO 8878 (including only the aspects of Addendum 1 contained in SC 6 N5181 dealing with 64 kbit/s throughput class, and the erratum contained in SC 6 N5185) except for the following exceptions:

IV.1 In Recommendatin X.223, the text in § 6.2.2.1.1 specifies that, under certain conditions, the NSAP Address is always carried in the AF whereas ISO 8878 leaves this as an option. ISO 8878 lists three conditions; Recommendation X.223 lists these three plus a fourth, as follows: "the NL entity, through local knowledge, is aware that the remote NL entity does not operate according to CCITT Recommendation X.223 and cannot recognize the AEF".

IV.2 In Recommendation X.223, the text in § 6.2.4 specifies that if "no use of Expedited Data" is indicated or if the NL entity cannot support 32-octet INTERRUPT packets, then the EDN facility is *always* omitted. for the same case, ISO 8878 specifies that the EDN facility either may be carried specifying "no use Expedited Data", or may be omitted.

IV.3 In § 6.2.5.1 (Throughput QOS Parameters) of Recommendation X.223, two new paragraphs have been added which are not present in ISO 8878. These paragraphs are the last paragraph in § 6.2.5.1.1, and the last paragraph in § 6.2.5.1.2.

Collectively, these paragraphs specify that whenever the Lowest Quality Acceptable sub-parameters of the Throughput QOS Parameters for both directions are "unspecified" in the N-CONNECT request, the MTCN facility is *not* included in the CALL REQUEST packet. ISO 8878 specifies that in such a case, the MTCN facility is encoded as 75 bits per second.

IV.4 In § 6.2.5.2 (Transit Delay QOS Parameter) of Recommendation X.223, four new paragraphs have been added which are not present in ISO 8878. These paragraphs are the last paragraph in § 6.2.5.2.1, the last paragraph in § 6.2.5.2.2, the last paragraph in § 6.2.5.2.3 and the last paragraph in § 6.2.5.2.4.

Collectively, these paragraphs specify that whenever the Target and Lowest Quality Acceptable sub-parameters of the Transit Delay QOS Parameter are "unspecified" in the N-CONNECT request, the EETDN facility is *not* included in the CALL REQUEST packet. ISO 8878 does not address the case of the EETDN facility being absent.

Additionally, in § 6.2.5.2.1 of Recommendation X.223, the last sentence in Item d specifies that in DTE-to-DTE operational environments the usage of the TDSAI facility is for further study. ISO 8878 does not have such a sentence.

IV.5 The scope of Recommendation X.223 does not include for provision of the OSI Connection-mode Network Service over 1980 X.25 subnetworks. Conversely, ISO 8878 provides for this and defines a protocol mechanism in Annex A. Also, material related to interoperability issues, including those raised by the presence of Annex A, is included in Annex B to ISO 8878 and is not included in this Recommendation.

Recommendation X.224

TRANSPORT PROTOCOL SPECIFICATION FOR OPEN SYSTEMS INTERCONNECTION FOR CCITT APPLICATIONS[1]

(Malaga-Torremolinos, 1984; amended at Melbourne, 1988)

The CCITT,

considering

(a) that Recommendation X.200 defines the Reference Model of Open Systems Interconnection for CCITT Applications;

(b) that Recommendation X.214 is the Transport Service Definition for Open Systems Interconnection for CCITT Applications;

(c) that Recommendation X.213 is the Network Service Definition for Open Systems Interconnection for CCITT Applications;

(d) that Recommendation T.70 defines the Network-Independent Basic Transport Service for Teletex,

unanimously declares

(1) that scope, field of application, references, definitions, symbols and abbreviations are given in §§ 1 to 4;

(2) that the Transport Protocol is overviewed in § 5;

(3) that the elements of procedure are as specified in § 6;

(4) that the classes of protocol are as specified in §§ 7 to 12;

(5) that the structure and encoding of the transport protocol data units is as specified in § 13;

(6) that the conformance requirements are as specified in § 14;

(7) that the state table description of the Transport Protocol is contained in Annex A.

CONTENTS

[1] Recommendation X.224 and ISO 8073 (Information Processing Systems – Open Systems Interconnection – Transport Protocol Specification) were developed in close collaboration and are technically aligned, except for the differences noted in Appendix II.

0 Introduction

The Transport Protocol is one of a set of Recommendations produced to facilitate the interconnection of computer systems. The set of Recommendations covers the services and protocols required to achieve such interconnection.

The Transport Protocol is positioned with respect to other related Recommendations by the layers defined in the Reference Model of Open Systems Interconnection for CCITT Applications [1]. It is most closely related to, and lies within the field of application of the Transport Service [2]. It also uses and makes reference to the Network Service [3], whose provisions it assumes in order to accomplish the transport protocol's aims. The interrelationship of these Recommendations is depicted in Figure 1/X.224.

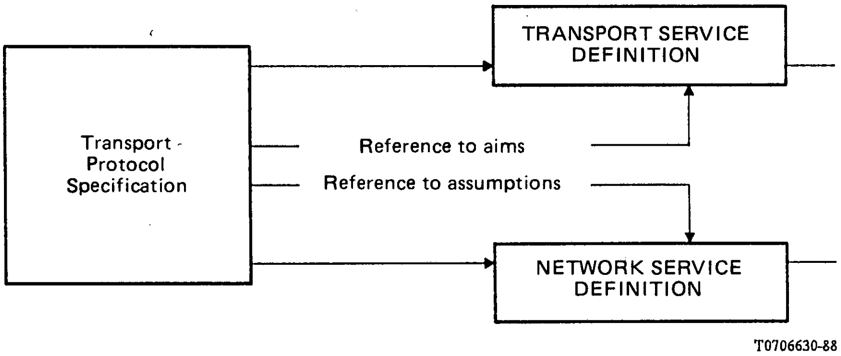

TRANSPORT SERVICE DEFINITION
Transport Protocol Specification
Reference to aims
Reference to assumptions
NETWORK SERVICE DEFINITION
T0706630-88

FIGURE 1/X.224

Relationship between the Transport Protocol and adjacent services

This Recommendation specifies a common encoding and a number of classes of transport protocol procedures to be used with different network qualities of service.

It is intended that the Transport Protocol should be simple but general enough to cater for the total range of Network Service qualities possible, without restricting future extensions.

The protocol is structured to give rise to classes of protocol which are designed to minimize possible incompatibilities and implementation costs.

The classes are selectable with respect to the Transport and Network Services in providing the required quality of service for the interconnection of two session entities (note that each class provides a different set of functions for enhancement of service qualities).

This protocol defines mechanisms that can be used to optimize network tariffs and enhance the following qualities of service:

a) different throughput;

b) different error rates;

c) integrity of data requirements;

d) reliability requirements.

It does not require an implementation to use all of these mechanisms, nor does it define methods for measuring achieved quality of service or criteria for deciding when to release transport connections following quality of service degradation.

The primary aim of this Recommendation is to provide a set of rules for communication expressed in terms of the procedures to be carried out by peer entities at the time of communication. These rules for communication are intended to provide a sound basis for development in order to serve a variety of purposes:

a) as a guide for implementors and designers;

b) for use in the testing and procurement of equipment;

c) as part of an agreement for the admittance of systems into the open systems environment;

d) as a refinement of the understanding of OSI.

It is expected that the initial users of the Recommendation will be designers and implementors of equipment and the Recommendation contains, in notes or in Annexes, guidance on the implementation of the procedures defined in the Recommendation.

It should be noted that, as the number of valid protocol sequences is very large, it is not possible with current technology to verify that an implementation will operate the protocol defined in this Recommendation correctly under all circumstances. It is possible by means of testing to establish confidence that an implementation correctly operates the protocol in a representative sample of circumstances. It is, however, intended that this Recommendation can be used in circumstances where two implementations fail to communicate in order to determine whether one or both have failed to operate the protocol correctly.

This Recommendation contains a section on conformance of equipment claiming to implement the procedures in this Recommendation. Attention is drawn to the fact that the Recommendation does not contain any tests to demonstrate this conformance.

The variations and options available within this Recommendation are essential to enable a Transport Service to be provided for a wide variety of applications over a variety of network qualities. Thus, a minimally conforming implementation will not be suitable for use in all possible circumstances. It is important therefore to qualify all references to this Recommendation with statements of the options provided or required or with statements of the intended purpose of provision or use.

1 Scope and field of application

1.1 This Recommendation specifies:

a) five classes of procedures:

1) Class 0: Simple Class;
2) Class 1: Basic Error Recovery Class;
3) Class 2: Multiplexing Class;

4) Class 3: Error Recovery and Multiplexing Class;
5) Class 4: Error Detection and Recovery Class;

for the connection oriented transfer of data and control information from one transport entity to a peer transport entity;

b) the means of negotiating the class of procedures to be used by the transport entities;

c) the structure and encoding of the transport protocol data units used for the transfer of data and control information.

1.2 The procedures are defined in terms of:

a) the interactions between peer transport entities through the exchange of transport protocol data units;

b) the interactions between a transport entity and the transport service user in the same system through the exchange of transport service primitives;

c) the interactions between a transport entity and the network service provider through the exchange of network service primitives.

These procedures are defined in the main text of the standard supplemented by state tables in Annex A.

1.3 These procedures are applicable to instances of communication between systems which support the Transport Layer of the OSI Reference Model and which wish to interconnect in an open systems environment.

1.4 This Recommendation specifies, in § 14, conformance for systems implementing these procedures. It does not contain tests which can be used to demonstrate these conformance requirements.

2 References

[1] Recommendation X.200 – Reference Model of Open Systems Interconnection for CCITT Applications (see also ISO 7498);

[2] Recommendation X.214 – Transport Service Definition for Open Systems Interconnection for CCITT Applications (see also ISO 8072;

[3] Recommendation X.213 – Network Service Definition for Open Systems Interconnection for CCITT Applications (see also ISO 8348;

[4] Recommendation T.70 – Network-Independent Basic Transport Service for Teletex;

3 Definitions

Note – The definitions contained in this section make use of abbreviations defined in § 4.

3.1 This Recommendation is based on the concepts developed in the Reference Model of Open Systems Interconnection for CCITT Applications [1] and makes use of the following terms defined in that Recommendation:

a) concatenation and separation;

b) segmenting and reassembling;

c) multiplexing and demultiplexing;

d) splitting and recombining;

e) flow control.

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

3.2.1 equipment

Hardware or software or a combination of both; it need not be physically distinct within a computer system.

3.2.2 **transport service user**

An abstract representation of the totality of those entities within a single system that make use of the transport service.

3.2.3 **network service provider**

An abstract machine which models the totality of the entities providing the network service, as viewed by a transport entity.

3.2.4 **local matter**

A decision made by a system concerning its behaviour in the Transport Layer that is not subject to the requirements of this protocol.

3.2.5 **initiator**

A transport entity that initiates a CR TPDU.

3.2.6 **responder**

A transport entity with whom an initiator wishes to establish a transport connection.

Note – Initiator and responder are defined with respect to a single transport connection. A transport entity can be both an initiator and responder simultaneously.

3.2.7 **sending transport entity**

A transport entity that sends a given TPDU.

3.2.8 **receiving transport entity**

A transport entity that receives a given TPDU.

3.2.9 **preferred class**

The protocol class that the initiator indicates in a CR TPDU as its first choice for use over the transport connection.

3.2.10 **alternative class**

A protocol class that the initiator indicates in a CR TPDU as an alternative choice for use over the transport connection.

3.2.11 **proposed class**

A preferred class or an alternative class.

3.2.12 **selected class**

The protocol class that the responder indicates in a CC TPDU that it has chosen for use over the transport connection.

3.2.13 **proposed parameter**

The value for a parameter that the initiator indicates in a CR TPDU that it wishes to use over the transport connection.

3.2.14 **selected parameter**

The value for a parameter that the responder indicates in a CC TPDU that it has chosen for use over the transport connection.

3.2.15 **error indication**

An N-RESET indication, or an N-DISCONNECT indication with a reason code indicating an error, that a transport entity receives from the NS-provider.

3.2.16 **invalid TPDU**

A TPDU which does not comply with the requirements of this Recommendation for structure and encoding.

3.2.17 **protocol error**

A TPDU whose use does not comply with the procedures for the class.

3.2.18 **sequence number**

a) The number in the TPDU-NR field of a DT TPDU which indicates the order in which the DT TPDU was transmitted by a transport entity.

b) The number in the YR-TU-NR field of an AK or RJ TPDU which indicates the sequence number of the next DT TPDU expected to be received by a transport entity.

3.2.19 **transmit window**

The set of consecutive sequence numbers which a transport entity has been authorised by its peer entity to send at a given time on a given transport connection.

3.2.20 **lower window edge**

The lowest sequence number in a transmit window.

3.2.21 **upper window edge**

The sequence number which is one greater than the highest sequence number in the transmit window.

3.2.22 **upper window edge allocated to the peer entity**

The value that a transport entity communicates to its peer entity to be interpreted as its new upper window edge.

3.2.23 **closed window**

A transmit window which contains no sequence number.

3.2.24 **window information**

Information contained in a TPDU relating to the upper and the lower window edges.

3.2.25 **frozen reference**

A reference which is not available for assignment to a connection because of the requirements of § 6.18.

3.2.26 **unassigned reference**

A reference that is neither currently in use for identifying a transport connection nor in a frozen state.

3.2.27 **transparent (data)**

TS-user data which is transferred intact between transport entities and which is unavailable for use by the transport entities.

3.2.28 **owner (of a network connection)**

The transport entity that issued the N-CONNECT request leading to the creation of that network connection.

3.2.29 **retained TPDU**

A TPDU which is subject to the retransmission procedure or retention until acknowledgment procedure and is available for possible retransmission.

4 Symbols and abbreviations

4.1 *Data units*

TPDU	Transport Protocol Data Unit
TSDU	Transport Service Data Unit
NSDU	Network Service Data Unit

4.2 *Types of transport protocol data unit*

CR TPDU	Connection Request TPDU
CC TPDU	Connection Confirm TPDU
DR TPDU	Disconnect Request TPDU
DC TPDU	Disconnect Confirm TPDU
DT TPDU	Data TPDU
ED TPDU	Expedited Data TPDU
AK TPDU	Data Acknowledge TPDU
EA TPDU	Expedited Acknowledge TPDU
RJ TPDU	Reject TPDU
ER TPDU	Error TPDU

4.3 *TPDU fields*

LI	Length Indicator (field)
CDT	Credit (field)
TSAP-ID	Transport Service Access Point Identifier (field)
DST-REF	Destination Reference (field)
SRC-REF	Source Reference (field)
EOT	End of TSDU mark
TPDU-NR	DT TPDU number (field)
ED-TPDU-NR	ED TPDU number (field)
YR-TU-NR	Sequence number response (field)
YR-EDTU-NR	ED TPDU number response (field)

4.4 *Times and associated variables*

T1	Local retransmission time
N	The maximum number of retransmissions
L	Time bound on reference and sequence numbers

I	Inactivity time
W	Window time
TTR	Time to try reassignment/resynchronization
TWR	Time to wait for reassignment/resynchronization
TS1	Supervisory timer 1
TS2	Supervisory timer 2
M_{LR}	NSDU lifetime local-to-remote
M_{RL}	NSDU lifetime remote-to-local
E_{LR}	Expected maximum transit delay local-to-remote
E_{RL}	Expected maximum transit delay remote-to-local
R	Persistence time
A_L	Local acknowledge time
A_R	Remote acknowledge time

4.5 *Miscellaneous*

TS-user	Transport Service user
TSAP	Transport Service Access Point
NS-provider	Network Service provider
NSAP	Network Service Access Point
QOS	Quality of service

5 Overview of the transport protocol

Note – This overview is not exhaustive and has been provided for guidance to the reader of this Recommendation.

5.1 *Service provided by the transport layer*

The protocol specified in this Recommendation supports the transport service defined in [2].

Information is transferred to and from the TS-user in the transport service primitives listed in Table 1/X.224.

5.2 *Service assumed from the network layer*

The protocol specified in this Recommendation assumes the use of the network services defined in [3].

Information is transferred to and from the NS-provider in the network service primitives listed in Table 2/X.224.

5.3 *Functions of the transport layer*

5.3.1 *Overview of functions*

The functions in the transport layer are those necessary to bridge the gap between the services available from the network layer and those to be offered to the TS-users.

The functions in the transport layer are concerned with the enhancement of quality of service, including aspects of cost optimization.

The functions are grouped below into those used at all times during a transport connection and those concerned with connection establishment, data transfer and release.

Note – This Recommendation does not include the following functions which are under consideration for inclusion in future editions of this Recommendation:

a) encryption;

b) accounting mechanisms;

c) status exchanges and monitoring of QOS;

d) blocking;

e) temporary release of network connections;

f) alternative checksum algorithm.

TABLE 1/X.224

Transport service primitives

Primitive	Parameters
T-CONNECT request T-CONNECT indication	Called Address, Calling Address, Expedited Data Option, Quality of Service, TS-User data.
T-CONNECT response T-CONNECT confirm	Responding Address, Quality of Service, Expedited Data Option, TS-User data.
T-DATA request T-DATA indication	TS-User data.
T-EXPEDITED DATA request T-EXPEDITED DATA indication	TU-User data.
T-DISCONNECT request	TS-User data.
T-DISCONNECT indication	Disconnect reason, TS-User data.

5.3.1.1 *Functions used at all times*

The following functions, depending on the selected class and options, are used at all times during a transport connection:

a) *transmission of TPDUs* (see §§ 6.2 and 6.9);

b) *multiplexing and demultiplexing* (see § 6.15), a function used to share a single network connection between two or more transport connections;

c) *error detection* (see §§ 6.10, 6.13 and 6.17), a function used to detect the loss, corruption, duplication, misordering or misdelivery of TPDUs;

d) *error recovery* (see §§ 6.12, 6.14, 6.18, 6.19, 6.20, 6.21 and 6.22), a function used to recover from detected and signalled errors.

TABLE 2/X.224

Network service primitives

Primitives	X/Y	Parameters (1, 2)	X/Y/Z
N-CONNECT request	X	Called address,	X
N-CONNECT indication	X	Calling address,	X
		Receipt confirmation selection,	Y
		Expedited data selection,	Y
		QOS parameter set,	X
		NS user data (3).	Z
N-CONNECT response	X	Responding address.	X
N-CONNECT confirmation	X	Receipt confirmation selection,	Y
		Expedited data selection,	Y
		QOS parameter set,	X
		NS user data (3).	Z
N-DATA request	X	NX-user data.	X
N-DATA indication	X	Confirmation request.	Y
N-DATA ACKNOWLEDGE request	Y		
N-DATA ACKNOWLEDGE indication	Y		
N-EXPEDITED DATA request	Y	NS user data.	Y
N-EXPEDITED DATA indication	Y		
N-RESET request	X	Reason.	Z
N-RESET indication	X	Originator,	Z
		Reason.	Z
N-RESET response	X		
N-RESET confirmation	X		
N-DISCONNECT request	X	Reason,	Z
		NS user data,	Z
		Responding address	Z
N-DISCONNECT indication	X	Originator,	Z
		Reason,	Z
		NS user data	Z
		Responding address.	Z

X: The Transport Protocol assumes that this feature is provided by all network service providers.

Y: The Transport Protocol assumes that this feature is provided by some network service providers and a mechanism is provided to optionally use the feature.

Z: The Transport Protocol does not use this parameter.

Note 1 – The parameters listed in this table are those in the network service definition (Reference 3).

Note 2 – The way the parameters are exchanged between the transport entity and the NS-provider is a local matter.

Note 3 – Although not used in the transport protocol *per se*, this parameter may be used for transport protocol identification, as specified in Annex B.

5.3.1.2 *Connection establishment*

The purpose of connection establishment is to establish a transport connection between two TS-users. The following functions of the transport layer during this phase match the TS-users' requested quality of service with the services offered by the network layer:

a) select network service which best matches the requirement of the TS-user taking into account charges for various services (see § 6.5);

b) decide whether to multiplex multiple transport connections onto a single network connection (see § 6.5);

c) establish the optimum TPDU size (see § 6.5);

d) select the functions that will be operational upon entering the data transfer phase (see § 6.5);

e) map transport addresses onto network addresses;

f) provide a means to distinguish between two different transport connections (see § 6.5);

g) transport of TS-user data (see § 6.5).

5.3.1.3 *Data transfer*

The purpose of data transfer is to permit duplex transmission of TSDUs between the two TS-users connected by the transport connection. This purpose is achieved by means of two-way simultaneous communication and by the following functions, some of which are used or not used in accordance with the result of the selection performed in connection establishment.

a) *concatenation and separation* (see § 6.4), a function used to collect several TPDUs into a single NSDU at the sending transport entity and to separate the TPDUs at the receiving transport entity;

b) *segmenting and reassembling* (see § 6.3), a function used to segment a single TSDU into multiple TPDUs at the sending transport entity and to reassemble them into their original format at the receiving transport entity;

c) *splitting and recombining* (see § 6.23), a function allowing the simultaneous use of two or more network connections to support the same transport connection;

d) *flow control* (see § 6.16), a function used to regulate the flow of TPDUs between two transport entities on one transport connection;

e) *transport connection identification,* a means to uniquely identify a transport connection between the pair of transport entities supporting the connection during the lifetime of the transport connection;

f) *expedited data* (see § 6.11), a function used to bypass the flow control of normal data TPDU. Expedited data TPDU flow is controlled by separate flow control;

g) *TSDU delimiting* (see § 6.3), a function used to determine the beginning and ending of a TSDU.

5.3.1.4 *Release*

The purpose of release (see §§ 6.7 and 6.8) is to provide disconnection of the transport connection, regardless of the current activity.

5.4 *Classes and options*

5.4.1 *General*

The functions of the transport layer have been organized into classes and options.

A class defines a set of functions. Options define those functions within a class which may or may not be used.

This Recommendation defines five classes of protocol:

a) Class 0: Simple Class;

b) Class 1: Basic Error Recovery Class;

c) Class 2: Multiplexing Class;

d) Class 3: Error Recovery and Multiplexing Class;

e) Class 4: Error Detection and Recovery Class.

Note 1 – Transport connections of Classes 2, 3 and 4 may be multiplexed together onto the same network connection.

Note 2 – Classes 0 to 3 do not specify mechanisms to detect unsignalled network transmission failures.

5.4.2 *Negotiation*

The use of classes and options is negotiated during connection establishment. The choice made by the transport entities will depend on:

a) the TS-users' requirements expressed via T-CONNECT service primitives;

b) the quality of the available network services;

c) the user required service versus cost ratio acceptable for the TS-user.

5.4.3 *Choice of network connection*

The following list classifies network services in terms of quality with respect to error behaviour in relation to user requirements; its main purpose is to provide a basis for the decision regarding which class of transport connection should be used in conjunction with a given network connection.

a) *Type A* – Network connection with acceptable residual error rate (for example not signalled by disconnect or reset) and acceptable rate of signalled errors.

b) *Type B* – Network connections with acceptable residual error rate (for example not signalled by disconnect or reset) but unacceptable rate of signalled errors.

c) *Type C* – Network connections with unacceptable residual error rate.

It is assumed that each transport entity is aware of the quality of service provided by particular Network connection.

5.4.4 *Characteristics of Class 0*

Class 0 provides the simplest type of transport connection and is fully compatible with Recommendation T.70 [4].

Class 0 has been designed to be used with type A network connections.

5.4.5 *Characteristics of Class 1*

Class 1 provides a basic transport connection with minimal overheads.

The main purpose of the class is to recover from network disconnect or reset.

Selection of this class is usually based on reliability criteria. Class 1 has been designed to be used with type B network connections.

5.4.6 *Characteristics of Class 2*

5.4.6.1 *General*

Class 2 provides a way to multiplex several transport connections onto a single network connection. This class has been designed to be used with type A network connections.

5.4.6.2 *Use of explicit flow control*

The objective is to provide flow control to help avoid congestion at transport-connection-end-points and on the network connection. Typical use is when traffic is heavy and continuous, or when there is intensive multiplexing. Use of flow control can optimize response times and resource utilization.

5.4.6.3 *Non-use of explicit flow control*

The objective is to provide a basic transport connection with minimal overheads suitable when explicit disconnection of the transport connection is desirable. The option would typically be used for unsophisticated terminals, and when no multiplexing onto network connections is required. Expedited data is never available.

5.4.7 *Characteristics of Class 3*

Class 3 provides the characteristics of Class 2 plus the ability to recover from network disconnect or reset. Selection of this class is usually based upon reliability criteria. Class 3 has been designed to be used with type B network connections.

5.4.8 *Characteristics of Class 4*

Class 4 provides the characteristics of Class 3, plus the capability to detect and recover from errors which occur as a result of the low grade of service available from the NS-provider. The kinds of errors to be detected include: TPDU loss, TPDU delivery out of sequence, TPDU duplication and TPDU corruption. These errors may affect control TPDUs as well as data TPDUs.

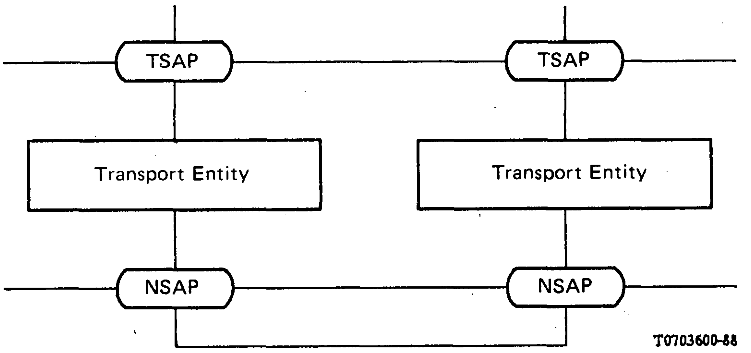
This class also provides for increased throughput capability and additional resilience against network failure.

Class 4 has been designed to be used with type C network connections.

5.5 *Model of the transport layer*

A transport entity communicates with its TS-users through one or more TSAPs by means of the service primitives as defined by the transport service definition (Reference 2). Service primitives will cause or be the result of transport protocol data unit exchanges between the peer transport entities supporting a transport connection. These protocol exchanges are effected using the services of the network layer as defined by the network service definition [3] through one or more NSAPs.

Transport connection endpoints are identified in end systems by an internal, implementation-dependent mechanism so that the TS-user and the transport entity can refer to each transport connection (see Figure 2/X.224).



Note – For the purpose of illustration, this figure shows only one TSAP and one NSAP for each transport entity. In certain instances, more than one TSAP and/or more than one NSAP may be associated with a particular transport entity.

FIGURE 2/X.224

Model of the transport layer

6 Elements of procedure

This section contains elements of procedure which are used in the specification of protocol classes in §§ 7 to 12. These elements are not meaningful on their own.

The procedures define the transfer of TPDUs whose structure and coding is specified in § 13. Transport entities shall accept and respond to any TPDU received in a valid NSDU and may issue TPDUs initiating specific elements of procedure specified in this section.

Note – Where network service primitives or TPDUs and parameters used are not significant for a particular element of procedure, they have not been included in the specification.

6.1 *Assignment to network connection*

6.1.1 *Purpose*

The procedure is used in all classes to assign transport connections to network connections.

6.1.2 *Network service primitives*

The procedure makes use of the following network service primitives:

a) N-CONNECT;

b) N-DISCONNECT.

6.1.3 *Procedure*

Each transport connection shall be assigned to a network connection. The initiator may assign the transport connection to an existing network connection of which it is the owner or to a new network connection (see Note 1) which it creates for this purpose.

The initiator shall not assign or reassign the transport connection to an existing network connection if the protocol class(es) proposed or the class in use for the transport connection are incompatible with the current usage of the network connection with respect to multiplexing (see Note 2).

During the resynchronization (see § 6.14) and reassignment after failure (see § 6.12) procedures, a transport entity may reassign a transport connection to another network connection joining the same NSAPs, provided that it is the owner of the network connection and that the transport connection is assigned to only one network connection at any given time.

During the splitting procedure (see § 6.23), a transport entity may assign a transport connection to any additional network connection joining the same NSAPs, provided that it is the owner of the network connection and that multiplexing is possible on the network connection.

The responder becomes aware of the assignment when it receives:

a) a CR TPDU during the connection establishment procedure (see § 6.5); or

b) an RJ TPDU or a retransmitted CR or DR TPDU during the resynchronization (see § 6.14) and reassignment after failure (see § 6.12) procedures; or

c) any TPDU when splitting (see § 6.23) is used.

Note 1 – When a new network connection is created, the quality of service requested is a local matter, although it will normally be related to the requirements of transport connection(s) expected to be assigned to it.

Note 2 – An existing network connection may also not be suitable if, for example, the quality of service requested for the transport connection cannot be attained by using or enhancing the network connection.

Note 3 – A network connection with no transport connection(s) assigned to it, may be available after initial establishment or because all of the transport connections previously assigned to it have been released. It is suggested that only the owner of such a network connection should release it. Furthermore, it is suggested that it not be released immediately after the transmission of the final TPDU of a transport connection – either a DR TPDU in response to a CR TPDU or a DC TPDU in response to a DR TPDU. An appropriate delay will allow the TPDU concerned to reach the other transport entity, allowing the freeing of any resources associated with the transport connection concerned.

Note 4 – After the failure of a network connection, transport connections which were previously multiplexed together may be assigned to different network connections, and vice versa.

Note 5 – The transport protocol identification procedures specified in Annex B may need to be considered in conjunction with this procedure.

6.2 *Transport protocol data unit (TPDU) transfer*

6.2.1 *Purpose*

The TPDU transfer procedure is used in all classes to convey transport protocol data units in user data fields of network service primitives.

6.2.2 *Network service primitives*

This procedure uses the following network service primitives:

a) N-DATA;

b) N-EXPEDITED DATA.

6.2.3 *Procedure*

The transport protocol data units (TPDUs) defined for the protocol are listed in § 4.2.

When the network expedited variant has been selected for Class 1, the transport entities shall transmit and receive ED and EA TPDUs as NS-user data parameters of N-EXPEDITED DATA primitives.

In all other cases, transport entities shall transmit and receive TPDUs as NS-user data parameters of N-DATA primitives.

When a TPDU is put into an NS-user data parameter, the significance of the bits within an octet and the order of octets within a TPDU shall be as defined in § 13.2.

Note 1 – TPDUs may be concatenated (see § 6.4).

Note 2 – The transport protocol identification procedures specified in Annex B may need to be considered in conjunction with this procedure.

6.3 *Segmenting and reassembling*

6.3.1 *Purpose*

The segmenting and reassembling procedure is used in all classes to map TSDUs onto TPDUs.

6.3.2 *TPDUs and parameters used*

The procedure makes use of the following TPDU and parameter:

DT TPDU:

– End of TSDU.

6.3.3 *Procedure*

A transport entity shall map a TSDU on to an ordered sequence of one or more DT TPDUs. This sequence shall not be interrupted by other DT TPDUs on the same transport connection.

All DT TPDUs except the last DT TPDU in a sequence greater than one shall have a length of data greater than zero.

Note 1 – The EOT of a DT TPDU indicates whether or not there are subsequent DT TPDUs in the sequence.

Note 2 – There is no requirement that the DT TPDUs shall be of the maximum length selected during connection establishment.

6.4 *Concatenation and separation*

6.4.1 *Purpose*

The procedure for concatenation and separation is used in Classes 1, 2, 3 and 4 to convey multiple TPDUs in one NSDU.

6.4.2 *Procedure*

A transport entity may concatenate TPDUs from the same or different transport connections while maintaining the order of TPDUs for a given transport connection compatible with the protocol operation.

A valid set of concatenated TPDUs may contain:

a) any number of TPDUs from the following list: AK, EA, RJ, ER, DC TPDUs provided that these TPDUs come from different transport connections;

b) no more than one TPDU from the following list: CR, DR, CC, DT, ED TPDUs; if this TPDU is present, it shall be placed last in the set of concatenated TPDUs.

A transport entity shall accept a valid set of concatenated TPDUs.

Note 1 – The TPDUs within a concatenated set may be distinguished by means of the length indicator parameter.

Note 2 – The end of a TPDU containing data is indicated by the termination of the NSDU.

Note 3 – The number of concatenated TPDUs referred to in § 6.4.2 a) is bounded by the maximum number of transport connections which are multiplexed together, except during assignment or reassignment.

6.5 *Connection establishment*

6.5.1 *Purpose*

The procedure for connection establishment is used in all classes to create a new transport connection.

6.5.2 *Network service primitives*

The procedure uses the following network service primitive:

N-DATA.

6.5.3 *TPDUs and parameters used*

The procedure uses the following TPDUs and parameters:

a) CR TPDU;
   - CDT;
   - DST-REF (set to zero);
   - SRC-REF;
   - CLASS and OPTIONS (preferred), i.e.:
     i) class,
     ii) use of extended format,
     iii) non-use of explicit flow control in Class 2;
   - calling TSAP-ID;
   - called TSAP-ID;
   - TPDU size (proposed);
   - version number;
   - protection parameter;
   - checksum;
   - additional option selection (preferred), i.e.:
     i) use of network expedited in Class 1,
     ii) use of receipt confirmation in Class 1,
     iii) non-use of checksums in Class 4,
     iv) use of transport expedited data transfer service;

- alternative protocol class(es);
- acknowledge time;
- throughput (proposed);
- residual error rate (proposed);
- priority (proposed);
- transit delay (proposed);
- reassignment time;
- user data.

b) CC TPDU;
   - CDT;
   - DST-REF;
   - SRC-REF;
   - CLASS and OPTIONS (selected);
   - calling TSAP-ID;
   - called TSAP-ID;
   - TPDU size (selected);
   - protection parameter;
   - checksum;
   - additional option selection (selected);
   - acknowledge time;
   - throughput (selected);
   - residual error rate (selected);
   - priority (selected);
   - transit delay (selected);
   - user data.

6.5.4 *Procedure*

A transport connection is established by means of one transport entity (the *initiator*) transmitting a CR TPDU to the other transport entity (the *responder*), which replies with a CC TPDU. Before sending the CR TPDU, the initiator assigns the transport connection being created to one (or more if the splitting procedure is being used) network connection(s). It is this set of network connections over which the TPDUs are sent.

Note – Even if the initiator assigns the transport connection to more than one network connection, all the CR TPDUs (if repeated) or DR TPDUs with DST-REF set to zero which are sent prior to the receipt of the CC TPDU, shall be sent on the same network connection, unless an N-DISCONNECT indication is received. (This is necessary because the remote entity may not support Class 4 and therefore may not recognize splitting.) If the initiator has made other assignments, it will use them only after receipt of a Class 4 CC TPDU (see also the splitting procedure § 6.23).

During this exchange, all information and parameters needed for the transport entities to operate shall be exchanged or negotiated.

Note 1 – The transport protocol identification procedures specified in Annex B may need to be considered in conjunction with this procedure.

Note 2 – Except in Class 4, it is suggested that the initiator start an optional timer TS1 at the time the CR TPDU is sent. This timer should be stopped when the connection is considered as accepted or refused or unsuccessful. If the timer expires, the initiator should reset or disconnect the network connection and, in Classes 1 and 3, freeze the reference (see § 6.18). For all other transport connection(s) multiplexed on the same network connection, the procedures for reset or disconnect as appropriate should be followed.

When an unexpected duplicated CR TPDU is received (with Class 4 as preferred class), it shall be ignored in Classes 0, 1, 2 and 3 and a CC TPDU shall be returned in Class 4.

After receiving the CC TPDU for a class which includes the procedure for retention until acknowledgment of TPDUs, the initiator shall acknowledge the CC TPDU as defined in Table 5/X.224 (see § 6.13).

When the network expedited variant of expedited data transfer (see § 6.11) has been agreed (possible in Class 1 only), the responder shall not send an ED TPDU before the CC TPDU is acknowledged.

The following information is exchanged:

a) *references* – Each transport entity chooses a reference to be used by the peer entity which is 16 bits long and which is arbitrary except for the following restrictions:

   1) it shall not already be in use or frozen (see § 6.18),
   2) it shall not be zero.

   This mechanism is symmetrical and provides identification of the transport connection independent of the network connection. The range of references used for transport connections, in a given transport entity, is a local matter.

b) *calling and called TSAPs-IDs* (optional) – Indicate the calling and called transport service access points. When either network address unambiguously defines the transport address, this information may be omitted.

c) *initial credit* – Only relevant for classes which include the explicit flow control function.

d) *user data* – Not available if Class 0 is the preferred class (see Note). Up to 32 octets in other classes.

   Note – If Class 0 is a valid response according to Table 3/X.224, inclusion of user data in the CR TPDU may cause the responding entity to refuse the connection (e.g. if it only supports Class 0).

e) *acknowledgment time* – Only in Class 4.

f) *checksum parameter* – Only in Class 4.

g) *protection parameter* – This parameter and its semantics are user defined.

TABLE 3/X.224

Valid responses corresponding to preferred and any alternative class proposed in the CR TPDU

Preferred class	Alternative class					
	0	1	2	3	4	none
0	not valid	not valid	not valid	not valid	not valid	class 0
1	class 1 or 0	class 1 or 0	not valid	not valid	not valid	class 1 or 0
2	class 2 or 0	not valid	class 2	not valid	not valid	class 2
3	class 3, 2 or 0	class 3, 2, 1 or 0	class 3 or 2	class 3 or 2	not valid	class 3 or 2
4	class 4, 2 or 0	class 4, 2, 1 or 0	class 4 or 2	class 4, 3 or 2	class 4 or 2	class 4 or 2

Note 1 – The valid responses indicated in the table result from both *explicit* negotiation, whereby each of the classes proposed is a valid response and *implicit* negotiation, whereby:

- if class 3 or 4 is proposed then class 2 is a valid response;
- if class 1 is proposed then class 0 is a valid response.

Note 2 – Negotiation from class 2 to class 1 and from any class to a higher-numbered class is not valid.

Note 3 – Redundant combinations of proposed classes can occur (e.g. due to the implicit negotiation rules). These are not considered as protocol errors.

The following negotiations take place:

h) *protocol class* – The initiator shall propose a preferred class and any number of alternative classes which permit a valid response as defined in Table 3/X.224. The initiator should assume when it sends the CR TPDU that its preferred class will be agreed to, and commence the procedures associated with that class, except that if Class 0 or Class 1 is an alternative class, multiplexing shall not commence until a CC TPDU selecting the use of Classes 2, 3 or 4 has been received.

   Note – This means, for example, that when the preferred class includes resynchronization (see § 6.14), the resynchronization will occur if a reset is signalled during connection establishment.

   The responder shall select one class defined in Table 3/X.224 as a valid response corresponding to the preferred class and to the class(es), if any, contained in the alternative class parameter of the CR TPDU. It shall indicate the selected class in the CC TPDU and shall follow the procedures for the selected class.

   If the preferred class is not selected, then on receipt of the CC TPDU, the initiator shall adjust its operation according to the procedures of the selected class.

i) *TPDU size* – The initiator may propose a maximum size for TPDUs, and the responder may accept this value or respond with any value between 128 and the proposed value in the set of values available for the class (see § 13.3.4 b)).

   Note – The length of the CR TPDU does not exceed 128 octets (see § 13.3).

j) *normal or extended format* – Either normal or extended is available. When extended is used, this applies to CDT, TPDU-NR, ED-TPDU-NR, YR-TU-NR and YR-EDTU-NR parameters.

k) *checksum selection* – This defines whether or not TPDUs of the connection are to include a checksum.

l) *quality of service parameters* – This defines the throughput, transit delay, priority, and residual error rate.

   Note – The transport service defines transit delay as requiring a previously stated average TSDU size as a basis for any specification. The protocol, as specified in § 13.3.4, l), uses a value of 128 octets. Conversion to and from specifications based upon some other value is a local matter.

m) *the non-use of explicit flow control* in Class 2.

n) *the use of network receipt confirmation and network expedited* when Class 1 is to be used.

o) *the use of expedited data transfer service* – This allows both TS-users to negotiate the use or non-use of the expedited data transport service as defined in the transport service definition [2].

The following information is set only in the CR TPDU:

p) *version number* – This defines the version of the transport protocol used for this connection.

q) *reassignment time parameter* – This indicates the time for which the initiator will persist in following the reassignment after failure procedure.

The negotiation rules for the options are such that the initiator may propose either to use or not to use the option. The responder may either accept the proposed choice or select an alternative choice as defined in Table 4/X.224.

When a parameter [which is valid for the proposed class(es)] is absent and a default value is defined in this Recommendation, this is equivalent to the presence of the parameter with the default value.

In Class 2, whenever a transport entity requests or agrees to the transport expedited data transfer service or to the use of extended formats, it shall request or agree (respectively) to the use of explicit flow control.

6.6 *Connection refusal*

6.6.1 *Purpose*

The connection refusal procedure is used in all classes when a transport entity refuses a transport connection in response to a CR TPDU.

TABLE 4/X.224

Negotiation of options during connection establishment

Option	Proposal made by the initiator	Valid selection by the responder
Transport expedited data transfer service (Classes 1, 2, 3, 4 only)	Yes No	Yes or No No
Use of receipt confirmation (Class 1 only)	Yes No	Yes or No No
Use of network expedited variant (Class 1 only)	Yes No	Yes or No No
Non-use of checksums (Class 4 only)	Yes No	Yes or No No
Non-use of explicit flow control (Class 2 only)	Yes No	Yes or No No
Use of extended format (Class 2, 3 4 only)	Yes No	Yes or No No

Note – Table 4/X.224 defines the procedures for negotiation of options. This negotiation has been designed such that if the initiator proposes the mandatory implementation option specified in § 14, the responder has to accept use of this option over the transport connection, except for the use of the transport expedited data transfer service which may be rejected by the TS-user. If the initiator proposes a non-mandatory implementation option, the responder is entitled to select use of the mandatory implementation option for use over the transport connection.

6.6.2 *TPDUs and parameters used*

The procedure makes use of the following TPDUs and parameters:

a) DR TPDU:
- SRC-REF;
- reason;
- user data.

b) ER TPDU:
- reject cause;
- invalid TPDU.

6.6.3 *Procedure*

If a transport connection cannot be accepted, the responder shall respond to the CR TPDU with a DR TPDU. The reason shall indicate why the connection was not accepted. The source reference field in the DR TPDU shall be set to zero to indicate an unassigned reference.

If a DR TPDU is received, the initiator shall regard the connection as released.

The responder shall respond to an invalid CR TPDU by sending an ER or DR TPDU. If an ER TPDU is received in response to a CR TPDU, the initiator shall regard the connection as released.

Note 1 – When the invalid CR TPDU can be identified as having Class 0 as the preferred class, it is suggested to respond with an ER TPDU. For all other invalid CR TPDUs, either an ER TPDU or DR TPDU may be sent.

Note 2 – If the optional supervisory timer TS1 has been set for this connection, then the initiator should stop the timer on receipt of the DR or ER TPDU.

6.7 *Normal release*

6.7.1 *Purpose*

The release procedure is used by a transport entity in order to terminate a transport connection. The implicit variant is used only in Class 0. The explicit variant is used in Classes 1, 2, 3 and 4.

Note 1 – When the implicit variant is used (i.e. in Class 0), the lifetime of the transport connection is directly correlated with the lifetime of the network connection.

Note 2 – The use of the explicit variant of the release procedure enables the transport connection to be released independently of the underlying network connection.

6.7.2 *Network service primitives*

The procedure makes use of the following network service primitives:

a) N-DISCONNECT (implicit variant only),

b) N-DATA.

6.7.3 *TPDUs and parameters used*

The procedure makes use of the following TPDUs and parameters:

a) DR TPDU:
   - reason;
   - user data;
   - SRC-REF;
   - DST-REF.

b) DC TPDU.

6.7.4 *Procedure for implicit variant*

In the implicit variant, either transport entity disconnects a transport connection by disconnecting the network connection to which it is assigned. When a transport entity receives an N-DISCONNECT indication, this should be considered as the release of the transport connection.

6.7.5 *Procedure for explicit variant*

When the release of a transport connection is to be initiated, a transport entity:

a) if it has previously sent or received a CC TPDU (see Note 1), shall:

   1) send a DR TPDU;

   2) discard all subsequently received TPDUs other than a DR or DC TPDU;

   3) consider the transport connection released on receipt of a DR or DC TPDU;

b) if a) is not applicable, it shall:

   1) for classes other than Class 4, wait for acknowledgment of the outstanding CR TPDU; if it receives a CC TPDU, it shall follow the procedure in § 6.7.5 a);

   2) for Class 4, either send a DR TPDU with a zero value in the DST-REF field, or follow the procedure in § 6.7.5 b) 1).

   In the former case, receipt of a CC TPDU specifying Class 4 will be ignored. Receipt of a CC TPDU with another class will be processed as follows: if the class is 0, the network connection shall be disconnected; otherwise, a DR TPDU with the DST-REF field set to the value of the SRC-REF field of the received CC TPDU shall be sent and the release procedure of the class is continued.

A transport entity that receives a DR TPDU shall:

c) if it has previously sent a DR TPDU for the same transport connection, consider the transport connection released;

d) if it has previously sent a CR TPDU that has not been acknowledged by a CC TPDU, consider the connection refused (see § 6.6);

If the SRC-REF is not zero, a DC TPDU shall be sent using the SRC-REF as the DST-REF.

Note – In this case, the DR TPDU has been associated regardless of its SRC-REF field (see § 6.9.4).

e) if c) and d) are not applicable, send a DC TPDU and consider the transport connection released. If the received DR has the DST-REF field set to zero, then a DC with SRC-REF set to zero shall be sent, regardless of the local reference. If the entity receiving such a DR TPDU has previously decided to negotiate down the class, this entity is always entitled to consider such a DR TPDU as spurious. Since no association has been made, the transport connection is not released at the responder side but the CC TPDU, when sent, will be answered by a DR TPDU (spurious CC TPDU).

Note 1 – This requirement ensures that the transport entity is aware of the remote reference for the transport connection.

Note 2 – When the transport connection is considered as released, the local reference is either available for re-use or is frozen (see § 6.18).

Note 3 – After the release of a transport connection, the network connection can be released or retained to enable its re-use for the assignment of other transport connections (see § 6.1).

Note 4 – Except in Class 4, it is suggested that if a transport entity does not receive acknowledgment of a DR TPDU within time TS2, it should either reset or disconnect the network connection, and freeze the reference when appropriate (see § 6.18). For all other transport connection(s) multiplexed on the same network connection, the procedures for reset or disconnect as appropriate should be followed.

Note 5 – When a transport entity is waiting for a CC TPDU before sending a DR TPDU and the network connection is reset or released, it should consider the transport connection released and, in classes other than Classes 0 and 2, freeze the reference (see § 6.18).

6.8 *Error release*

6.8.1 *Purpose*

This procedure is used only in Classes 0 and 2 to release a transport connection on the receipt of a N-DISCONNECT or N-RESET indication.

6.8.2 *Network service primitives*

The procedure makes use of the following service primitives:

a) N-DISCONNECT indication;

b) N-RESET indication.

6.8.3 *Procedure*

When, on the network connection to which a transport connection is assigned, an N-DISCONNECT or N-RESET indication is received, both transport entities shall consider that the transport connection is released, and so inform the TS-users.

Note – In other classes, since error recovery is used, the receipt of an N-RESET indication or N-DISCONNECT indication will result in the invocation of the error recovery procedure.

6.9 *Association of TPDUs with transport connections*

6.9.1 *Purpose*

This procedure is used in all classes to interpret a received NSDU as TPDU(s) and, if possible, to associate each such TPDU with a transport connection.

6.9.2 *Network service primitives*

This procedure makes use of the following network service primitives:

a) N-DATA indication;

b) N-EXPEDITED DATA indication.

6.9.3 *TPDUs and parameters used*

This procedure makes use of the following TPDUs and parameters:

a) in any TPDU except: CR TPDU; DT TPDU in Classes 0 or 1; AK TPDU in Class 1:
   - DST-REF.

b) CR, CC, DR and DC TPDUs:
   - SRC-REF.

c) DT TPDU in Classes 0 or 1 and AK TPDU in Class 1.

6.9.4 *Procedures*

6.9.4.1 *Identification of TPDUs*

If the received NSDU or expedited NSDU cannot be decoded (i.e. does not contain one or more correct TPDUs) or is corrupted (i.e. contains a TPDU with a wrong checksum) then the transport entity shall:

a) if the network connection on which the error is detected has a Class 0 or Class 1 transport connection assigned to it, then treat as a protocol error (see § 6.22) for that transport connection;

b) otherwise:

   1) if the NSDU can be decoded but contains corrupted TPDUs, discard the TPDUs (Class 4 only) and optionally apply § 6.9.4.1 b) 2);

   2) if the NSDU cannot be decoded, issue an N-RESET (or N-DISCONNECT) request for the network connection and for all of the transport connections assigned to this network connection (if any), apply the procedures defined for handling of network signalled reset or disconnect.

If the NSDU can be decoded and is not corrupted, the transport entity shall:

c) if the network connection on which the NSDU was received has a Class 0 transport connection assigned to it, then consider the NSDU as forming one TPDU and associate the TPDU with the transport connection (see § 6.9.4.2);

d) otherwise, invoke the separation procedures and for each of the individual TPDUs in the order in which they appear in the NSDU apply the procedure defined in § 6.9.4.2.

6.9.4.2 *Association of individual TPDUs*

If the received TPDU is a CR TPDU, then, if it is a duplicate as recognized by using the NSAPs of the network connection, and the SRC-REF parameter, then it is associated with the transport connection created by the original copy of the CR TPDU; otherwise, it is processed as requesting the creation of a new transport connection.

If the received TPDU is a DT TPDU and the network connection has a Class 0 or Class 1 transport connection assigned to it, or an AK TPDU where a Class 1 transport connection is assigned, then the TPDU is associated with the transport connection.

Otherwise, the DST-REF parameter of the TPDU is used to identify the transport connection. The following cases are distinguished:

a) If the DST-REF is not allocated to a transport connection, the transport entity shall respond on the same network connection with a DR TPDU if the TPDU is a CC TPDU, with a DC TPDU if the TPDU is a DR TPDU and shall discard the TPDU if neither a DR TPDU nor CC TPDU. No association with a transport connection is made.

   Note – If the DR TPDU is carrying an SRC-REF field set to zero, then no DC TPDU shall be sent.

b) If the DST-REF is allocated to a connection, but the TPDU is received on a network connection to which the connection has not been assigned, then there are three cases:

   1) if the transport connection is of Class 4 and if the TPDU is received on a network connection with the same pair of NSAPs as that of the CR TPDU, then the TPDU is associated with this transport connection and considered as performing assignment;

   2) if the transport connection is not assigned to any network connection (waiting for reassignment after failure) and if the TPDU is received on a network connection with the same pair of NSAPs as that of the CR TPDU, then the association with that transport connection is made except in the case of DC, DR and CC TPDUs which are respectively described in § 6.9.4.2 c), d) and e);

   3) otherwise, the TPDU is considered as having a DST-REF not allocated to a transport connection [case a)].

c) If the TPDU is a DC TPDU, then it is associated with the transport connection to which the DST-REF is allocated, unless the SRC-REF is not the expected one, in which case the DC TPDU is discarded.

d) If the TPDU is a DR TPDU then there are four cases:

   1) if the SRC-REF is not as expected, then a DC TPDU with a DST-REF equal to the SRC-REF of the received DR TPDU is sent back and no association is made;

   2) if a CR TPDU is unacknowledged, then the DR TPDU is associated with the transport connection, regardless of the value of its SRC-REF parameter;

   3) if the transport entity implements Class 4 and if the DST-REF is zero and there is an unacknowledged CC TPDU or T-CONNECT response is awaited, then the DR TPDU shall be associated with the transport connection holding the SRC-REF as the remote reference;

   4) otherwise, the DR TPDU is associated with the transport connection identified by the DST-REF parameter.

e) If the TPDU is a CC TPDU whose DST-REF parameter identifies an open connection (one for which a CC TPDU has been previously received), and the SRC-REF in the CC TPDU does not match the remote reference, then a DR TPDU is sent back with DST-REF equal to the SRC-REF of the received CC TPDU and no association is made.

f) If none of the above cases apply, then the TPDU is associated with the transport connection identified by the DST-REF parameter.

6.10 *Data TPDU numbering*

6.10.1 *Purpose*

Data TPDU numbering is used in Classes 1, 2 (except when the non-use of explicit flow control option is selected), 3 and 4. Its purpose is to enable the use of recovery, flow control and resequencing functions.

6.10.2 *TPDUs and parameters used*

This procedure makes use of the following TPDU and parameter:

DT TPDU;

- TPDU-NR.

6.10.3 *Procedure*

A transport entity shall allocate the sequence number zero to the TPDU-NR of the first DT TPDU which it transmits for a transport connection. For subsequent DT TPDUs sent on the same transport connection, the transport entity shall allocate a sequence number one greater than the previous one.

When a DT TPDU is retransmitted, the TPDU-NR parameter shall have the same value as in the first transmission of that DT TPDU.

Modulo 2^7 arithmetic shall be used when normal formats have been selected and modulo 2^{31} arithmetic shall be used when extended formats have been selected. In this Recommendation, the relationships "greater than" and "less than" apply to a set of contiguous TPDU numbers whose range is less than the modulus and whose starting and finishing numbers are known. The term "less than" means "occurring sooner in the window sequence" and the term "greater than" means "occurring later in the window sequence".

6.11 *Expedited data transfer*

6.11.1 *Purpose*

Expedited data transfer procedures are selected during connection establishment. The network normal data variant may be used in Classes 1, 2, 3 and 4. The network expedited variant is only used in Class 1.

6.11.2 *Network service primitives*

The procedure makes use of the following network service primitives:

a) N-DATA;

b) N-EXPEDITED DATA.

6.11.3 *TPDUs and parameters used*

The procedure makes use of the following TPDUs and parameters:

a) ED TPDU:
   - ED-TPDU-NR.

b) EA TPDU:
   - YR-EDTU-NR.

6.11.4 *Procedure*

The TS-user data parameter of each T-EXPEDITED DATA request shall be conveyed as the data field of an Expedited Data (ED) TPDU.

Each ED TPDU received shall be acknowledged by an Expedited Acknowledge (EA) TPDU.

No more than one ED TPDU shall remain unacknowledged at any time for each direction of a transport connection.

An ED TPDU with a zero length data field shall be treated as a protocol error.

Note 1 – The network normal data variant is used, except when the network expedited variant (available in Class 1 only), has been agreed, in which case ED and EA TPDUs are conveyed in the data fields of N-EXPEDITED DATA primitives (see § 6.2.3).

Note 2 – No TPDUs can be transmitted using network expedited until the CC TPDU becomes acknowledged, to prevent the network expedited from overtaking the CC TPDU.

6.12 *Reassignment after failure*

6.12.1 *Purpose*

The reassignment after failure procedure is used in Classes 1 and 3 to commence recovery from an NS-provider signalled disconnect.

6.12.2 *Network service primitives*

The procedure uses the following network service primitive:

N-DISCONNECT indication.

6.12.3 *Procedure*

When an N-DISCONNECT indication is received for the network connection to which a transport connection is assigned, the initiator shall apply one of the following alternatives:

a) if the TTR timer has not already run out and no DR TPDU is retained, then:

1) assign the transport connection to a different network connection (see § 6.1) and start its TTR timer if not already started;

2) while waiting for the completion of assignment if:

- an N-DISCONNECT indication is received, repeat the procedure from § 6.12.3 a),
- the TTR timer expires, begin procedure § 6.12.3 b);

3) when reassignment is completed, begin resynchronization (see § 6.14) and:

- if a valid TPDU is received as the result of the resynchronization, stop the TTR timer, or
- if TTR runs out, wait for the next event, or
- if an N-DISCONNECT indication is received, then begin either procedure § 6.12.3 a) or § 6.12.3 b) depending on the TTR timer.

Note – After TTR expires and while waiting for the next event, it is suggested that the initiator set a timer with a value equal to TWR. If this timer expires before the next event, the initiator should begin the procedure in § 6.12.3 b);

b) if the TTR timer has run out, consider the transport connection as released and freeze the reference (see § 6.18);

c) if a DR TPDU is retained and the TTR timer has not run out, then follow the actions in either § 6.12.3 a) or § 6.12.3 b).

The responder shall start its TWR timer if not already started. The arrival of the first TPDU related to the transport connection (because of resynchronization by the initiator) completes the reassignment after failure procedure. The TWR timer is stopped and the responder shall continue with resynchronization (see § 6.14). If reassignment does not take place within this time, the transport connection is considered released and the reference is frozen (see § 6.18).

If reassignment occurs successfully, both transport entities shall continue with resynchronization.

Note – The transport protocol identification procedures specified in Annex B may need to be considered in conjunction with this procedure.

6.12.4 *Timers*

The reassignment after failure procedure uses two timers:

a) TTR, the time to try reassignment/resynchronization timer;

b) TWR, the time to wait for reassignment/resynchronization timer.

The TWR timer is used by the initiator. Its value shall not exceed two minutes minus the sum of the maximum disconnect propagation delay and the maximum transit delay of the network connections (see Note 1). The value for the TTR timer may be indicated in the CR TPDU.

The TWR timer is used by the responder. If the reassignment time parameter is present in the CR TPDU, the TWR timer value shall be greater than the sum of the TTR timer plus the maximum disconnect propagation delay plus the maximum transit delay of the network connections.

If the reassignment time parameter is not present in the CR TPDU, a default value of 2 minutes shall be used for the TWR timer.

Note 1 – Provided that the required quality of service is met, TTR may be set to zero (i.e., no reassignment), for example if the rate of NS-provider generated disconnects is very low.

Note 2 – Inclusion of the reassignment time parameter in the CR TPDU allows the responder to use a TWR value of less than 2 minutes.

Note 3 – If the optional TS1 and TS2 timers are used, it is suggested:

a) to stop TS1 or TS2 if running when TTR or TWR is started;

b) to restart TS1 or TS2 if necessary when the corresponding TPDU (CR TPDU or DR TPDU respectively) is repeated;

c) to select for TS1 and TS2 values greater than TTR.

6.13 *Retention until acknowledgment of TPDUs*

6.13.1 *Purpose*

The retention until acknowledgment of TPDUs procedure is used in Classes 1, 3 and 4 to enable and minimize retransmission after possible loss of TPDUs.

The confirmation of receipt variant is used only in Class 1 when it has been agreed during connection establishment (see Note).

The AK variant is used in Classes 3 and 4 and also in Class 1 when the confirmation of receipt variant has not been agreed during connection establishment.

Note – Use of confirmation of receipt variant depends on the availability of the network layer receipt confirmation service and the expected cost reduction.

6.13.2 *Network service primitives*

The procedure uses the following network service primitives:

a) N-DATA;

b) N-DATA ACKNOWLEDGE.

6.13.3 *TPDUs and parameters used*

The procedure uses the following TPDUs and parameters:

a) CR, CC, DR and DC TPDUs.

b) RJ and AK TPDUs:
   - YR-TU-NR.

c) DT TPDU:
   - TPDU-NR.

d) ED TPDU:
   - ED-TPDU-NR.

e) EA TPDU:
   - YR-EDTU-NR.

6.13.4 *Procedure*

Copies of the following TPDUs shall be retained upon transmission to permit their later retransmission:

CR, CC, DR, DT and ED TPDUs

except that if a DR TPDU is sent in response to a CR TPDU, there is no need to retain a copy of the DR TPDU.

A copy of each of these TPDUs shall be retained until:

a) it is acknowledged, as specified in Table 5/X.224; or

b) the transport connection is released.

In the confirmation of receipt variant, applicable only in Class 1, transport entities shall:

a) set the confirmation request parameter only if the data parameter contains a CC or DT TPDU (see Notes 1 and 2); and

b) issue an N-DATA ACKNOWLEDGE request when it receives an N-DATA indication with the confirmation request parameter set.

Note 1 – It is a local matter for each transport entity to decide which N-DATA requests should have the confirmation request parameter set. This decision will normally be related to the amount of storage available for retained copies of the DT TPDUs.

Note 2 – Use of the confirmation request parameter may affect the quality of network service.

TABLE 5/X.224

Acknowledgement of TPDUs

Retained TPDU	Variant	Retained until acknowledged by
CR	both	CC, DR or ER TPDU.
DR	both	DC or DR (in case of collision) TPDU.
CC	Confirmation of receipt	N-DATA ACKNOWLEDGE indication, RJ, DT, ED or EA TPDU.
CC	AK	RJ, DT, AK, ED or EA TPDU.
DT	Confirmation of receipt	N-DATA ACKNOWLEDGE indication corresponding to an N-DATA request which conveyed, or came after, the DT TPDU.
DT	AK	AK or RJ TPDU for which the YR-TU-NR is greater than TPDU-NR in the DT TPDU.
ED	both	EA TPDU for which the YR-EDTU-NR is equal to the ED-TPDU-NR in the ED TPDU.

6.14 *Resynchronization*

6.14.1 *Purpose*

The resynchronization procedures are used in Classes 1 and 3 to restore the transport connection to normal after a reset or during reassignment after failure according to § 6.12.

6.14.2 *Network service primitives*

The procedure makes use of the following network service primitive:

N-RESET indication.

6.14.3 *TPDUs and parameters used*

The procedure uses the following TPDUs and parameters:

a) CR, DR, CC and DC TPDUs.

b) RJ TPDU:
   - YR-TU-NR.

c) DT TPDU:
   - TPDU-NR.

d) ED TPDU:
   - ED-TPDU-NR.

e) EA TPDU:
   - YR-EDTU-NR.

6.14.4 *Procedure*

A transport entity which is notified of the occurrence of an N-RESET or which is performing reassignment after failure according to § 6.12 shall carry out the active resynchronization procedures (see § 6.14.4.1) *unless* any of the following hold:

a) the transport entity is the responder. In this case, the passive resynchronization procedure shall be carried out (see § 6.14.4.2);

b) the transport entity has elected not to reassign (see § 6.12.3 c)). In this case, no resynchronizaion takes place.

6.14.4.1 *Active resynchronization procedures*

The transport entity shall carry out one of the following actions:

a) if the TTR timer has been previously started and has run out (i.e. no valid TPDU has been received), the transport connection is considered as released and the reference is frozen (see § 6.18);

b) otherwise, the TTR timer shall be started (unless it is already running) and the first applicable one of the following actions shall be taken:

   1) if a CR TPDU is unacknowledged, then the transport entity shall retransmit it;

   2) if a DR TPDU is unacknowledged, then the transport entity shall retransmit it;

   3) otherwise, the transport entity shall carry out the data resynchronization procedures (§ 6.14.4.3).

   The TTR timer is stopped when a valid TPDU is received.

6.14.4.2 *Passive resynchronization procedures*

The transport entity shall not send any TPDUs until a TPDU has been received. The transport entity shall start its TWR timer if it was not already started (due to a previous N-DISCONNECT or N-RESET indication). If the timer runs out prior to the receipt of a valid TPDU which commences resynchronization (i.e. CR or DR or ED or RJ TPDU), the transport connection is considered as released and the reference is released (see § 6.18).

When a valid TPDU is received, the transport entity shall stop its TWR timer and carry out the appropriate one of the following actions, depending on the TPDU:

a) if it is a DR TPDU, then the transport entity shall send a DC TPDU;

b) if it is a repeated CR TPDU (see Note 1) then the transport entity shall carry out the action which is appropriate from the following:

   1) if a CC TPDU has alredy been sent, and acknowledged: treat as a protocol error;

   2) if a DR TPDU is unacknowledged (whether or not a CC TPDU is unacknowledged): retransmit the DR TPDU, but setting the source reference to zero;

   3) if the T-CONNECT response has not yet been received from the user: take no action;

   4) otherwise: retransmit the CC TPDU followed by any unacknowledged ED TPDU (see Note 2) and any DT TPDU.

   Note 1 – A repeated CR can be identified by being on a network connection with the appropriate network addresses and having a correct source reference.

   Note 2 – The transport entity should not use network expedited until the CC is acknowledged (see § 6.5). This rule prevents the network expedited from overtaking the CC TPDU;

c) if it is an RJ or ED TPDU, then one of the following actions shall be taken:

   1) if a DR TPDU is unacknowledged, then the transport entity shall retransmit it;

   2) if a CC TPDU is unacknowledged, the RJ or ED TPDU shall be considered as acknowledging the CC TPDU, and the transport entity shall carry out the data resynchronization procedures (§ 6.14.4.3);

   3) if a CC TPDU was never sent, the RJ or ED TPDU should be considered as a protocol error;

   4) otherwise, the transport entity shall carry out the data resynchronization procedures (§ 6.14.4.3.)

6.14.4.3 *Data resynchronization procedures*

The transport entity shall carry out the following actions in the following order:

a) (re)transmit any ED TPDU which is unacknowledged;

b) transmit an RJ TPDU with YR-TU-NR field set to the TPDU-NR of the next expected DT TPDU;

c) wait for the next TPDU from the other transport entity, unless it has already been received. If a DR TPDU is received, the transport entity shall send a DC TPDU, freeze the reference, inform the TS-user of the disconnection and take no further action [i.e. it shall not follow the procedures in § 6.14.4.3 d)]. If an RJ TPDU is received, the procedures of § 6.14.4.3 d) shall be followed. If an ED TPDU is received, the procedures as described in § 6.11 shall be followed. If it is a duplicated ED TPDU, the transport entity shall acknowledge it with an EA TPDU, discard the duplicated ED TPDU and wait again for the next TPDU;

d) (re)transmit any DT TPDUs which are unacknowledged, subject to any applicable flow control procedures (see Note).

   Note – The RJ TPDU may have reduced the credit.

6.15 *Multiplexing and demultiplexing*

6.15.1 *Purpose*

The multiplexing and demultiplexing procedures are used in Classes 2, 3 and 4 to allow several transport connections to share a network connection at the same time.

6.15.2 *TPDUs and parameters used*

The procedure makes use of the following TPDUs and parameters:

CC, DR, DC, DT, AK, ED, EA, RJ and ER TPDUs:

- DST-REF.

6.15.3 *Procedure*

The transport entities shall be able to send and receive on the same network connection TPDUs belonging to different transport connections.

Note 1 – When performing demultiplexing, the transport connection to which the TPDUs apply is determined by the procedures defined in § 6.9.

Note 2 – Multiplexing allows the concatenation of TPDUs belonging to different transport connections to be transferred in the same N-DATA primitive (see § 6.4).

6.16 *Explicit flow control*

6.16.1 *Purpose*

The explicit flow control procedure is used in Classes 2, 3 and 4 to regulate the flow of DT TPDUs independently of the flow control in the other layers.

6.16.2 *TPDUs and parameters used*

The procedure makes use of the following TPDUs and parameters:

a) CR, CC, AK and RJ TPDUs:
   - CDT.

b) DT TPDU:
   - TPDU-NR.

c) AK TPDU:
   - YR-TU-NR;
   - subsequence number;
   - flow control confirmation.

d) RJ TPDU:
   - YR-TU-NR.

6.16.3 *Procedure*

The procedures differ in different classes. They are defined in the sections specifying the separate classes.

6.17 *Checksum*

6.17.1 *Purpose*

The checksum procedure is used to detect corruption of TPDUs by the NS-provider.

Note – Although a checksum algorithm has to be adapted to the type of errors expected on the network connection, at present only one algorithm is defined.

6.17.2 *TPDUs and parameters used*

The procedure uses the following TPDUs and parameters:

All TPDUs:

- checksum.

6.17.3 *Procedure*

The checksum is used only in Class 4. It shall always be used for the CR TPDU, and shall be used for all other TPDUs unless the non-use of the checksum was selected during connection establishment.

The sending transport entity shall transmit TPDUs with the checksum parameter set such that the following formulae are satisfied:

$$\sum_{i=1}^{L} a_i \equiv 0 \text{ (modulo 255)}$$

$$\sum_{i=1}^{L} ia_i \equiv 0 \text{ (modulo 255)}$$

where

i = number (i.e. position) of an octet within the TPDU (see § 13.2).

a_i = value of octet in position i.

L = length of TPDU in octets.

A transport entity which receives a TPDU for a transport connection for which the use of checksum has been agreed and which does not satisfy the above formulae shall discard the TPDU (see also Note 2).

When a spurious TPDU is received and an answer is to be sent the transport entity shall:

a) if it supports the checksum algorithm and the received TPDU contains a checksum parameter, include a checksum parameter in the answering TPDU; or

b) in all other cases, not include a checksum parameter in the answering TPDU.

An entity not supporting checksum may always suppose that a CR TPDU with Class 4 proposed is correct and therefore negotiate down to a class lower than 4.

Note 1 – An efficient algorithm for determining the checksum parameters is given in Appendix I.

Note 2 – If the checksum is incorrect, it is not possible to know with certainty to which transport connection the TPDU is related; thus, further action may be taken for all the transport connections assigned to the network connection (see § 6.9).

Note 3 – The checksum proposed is easy to calculate and so will not impose a heavy burden on implementations. However, it will not detect insertion or loss of leading or trailing zeros and will not detect some octets misordering.

Note 4 – When a TPDU is received on a network connection, it is never possible to know with certainty that only Class 4 transport connections use this network connection because it may be a TPDU performing reassignment.

Therefore the only way to check the validity is the following:

a) if the network connection is used by a Class 0 or Class 1 transport connection, there is no checksum;

b) examine the TPDU code;

c) deduce the fixed part length;

d) from LI, deduce the variable part;

e) go through parameters and if the checksum parameter is found, then verify it;

f) if it is incorrect, then assume that transport connection is Class 4 and drop it;

g) if it is correct, then associate the TPDU with a transport connection; if the transport connection uses the checksum, it is correct; else, it must be considered as a protocol error.

6.18 *Frozen references*

6.18.1 *Purpose*

This procedure is used in order to prevent re-use of a reference while TPDUs associated with the old use of the reference may still exist.

6.18.2 *Procedure*

When a transport entity determines that a particular connection is released, it shall place the reference which it has allocated to the connection in a frozen state according to the procedures of the class. While frozen, the reference shall not be re-used.

Note – The frozen reference procedure is necessary because retransmission or misordering can cause TPDUs bearing a reference to arrive at an entity after it has released the connection for which it allocated the reference. Retransmission, for example, can arise when the class includes either resynchronization (see § 6.14) or retransmission on timeout (see § 6.19).

6.18.2.1 *Procedure for Classes 0 and 2*

This Recommendation does not specify frozen reference procedures for classes 0 and 2.

Note – For consistency with the other classes, references may be frozen as a local matter.

6.18.2.2 *Procedure for Classes 1 and 3*

The frozen reference procedure is used except in the following cases (see Note 1):

a) when the transport entity receives a DC TPDU in response to a DR TPDU which it has sent (see Note 2);

b) when the transport entity sends a DR TPDU in response to a CR TPDU which it has received (see Note 3);

c) when the transport entity has considered the connection to be released after the expiration of the TWR timer (see Note 4);

d) when the transport entity receives a DR or ER TPDU in response to a CR TPDU which it has sent.

The period of time for which the reference remains frozen shall be greater than the TWR time.

Note 1 – However, even in these cases, for consistency, freezing the reference may be done as a local decision.

Note 2 – When the DC TPDU is received, it is certain that the other transport entity considers the connection released.

Note 3 – When the DR or ER TPDU is sent, the peer transport entity has not been informed of any reference assignment and thus cannot possibly make use of a reference (this includes the case where a CC TPDU was sent, but was lost).

Note 4 – In § 6.18.2 c), the transport entity has already effectively frozen the reference for an adequate period.

6.18.2.3 *Procedure for Class 4*

The frozen reference procedure shall be used in Class 4. The period for which the reference remains frozen shall be greater than *L* (see § 12.2.1.1.6).

6.19 *Retransmission on timeout*

6.19.1 *Purpose*

The procedure is used in Class 4 to cope with unsignalled loss of TPDUs by the NS-provider.

6.19.2 *TPDUs used*

The procedure makes use of the following TPDUs:

CR, CC, DR, DT, ED and AK TPDUs.

6.19.3 *Procedure*

The procedure is specified in the procedures for Class 4 (see § 12.2.1.2 i)).

6.20 *Resequencing*

6.20.1 *Purpose*

The resequencing procedure is used in Class 4 to cope with misordering of TPDUs by the NS-provider.

6.20.2 *TPDUs and parameters used*

The procedure uses the following TPDUs and parameters:

a) DT TPDU:
   - TPDU-NR.

b) ED TPDU:
   - ED-TPDU-NR.

6.20.3 *Procedure*

The procedure is specified in the procedures for Class 4 (see § 12.2.3.5).

6.21 *Inactivity control*

6.21.1 *Purpose*

The inactivity control procedure is used in Class 4 to cope with unsignalled termination of a network connection.

6.21.2 *Procedure*

The procedure is specified in the procedures for Class 4 (see § 12.2.3.3).

6.22 *Treatment of protocol errors*

6.22.1 *Purpose*

The procedure for treatment of protocol errors is used in all classes to deal with invalid TPDUs.

6.22.2 *TPDUs and parameters used*

The procedure uses the following TPDUs and parameters:

a) ER TPDU:
   - reject cause;
   - invalid TPDU.

b) DR TPDU:
   - reason code.

6.22.3 *Procedure*

A transport entity that receives a TPDU that can be associated to a transport connection and is invalid or constitutes a protocol error (see §§ 3.2.16 and 3.2.17) shall take one of the following actions so as not to jeopardize any other transport connections not assigned to that network connection:

a) transmitting an ER TPDU;

b) resetting or closing the network connection; or

c) invoking the release procedures appropriate to the class.

Under certain circumstances it is also possible to discard the TPDU.

If an ER TPDU is sent in Class 0, it shall contain the octets of the invalid TPDU up to and including the octet where the error was detected (see Notes 3, 4 and 5).

If the TPDU cannot be associated with a particular transport connection, the transport entity shall follow the procedure in § 6.9.

Note 1 – In general, no further action is specified for the receiver of the ER TPDU, but it is suggested that it initiates the release procedure appropriate to the class. If the ER TPDU has been received as an answer to a CR TPDU, then the connection is regarded as released (see § 6.6).

Note 2 – Care should be taken by a transport entity receiving several invalid TPDUs or ER TPDUs to avoid looping if the error is generated repeatedly.

Note 3 – If the invalid received TPDU is greater than the selected maximum TPDU size, it is possible that it cannot be included in the invalid TPDU parameter of the ER TPDU.

Note 4 – It is suggested that the sender of the ER TPDU start a timer TS2 to ensure the release of the connection. If the timer expires, the transport entity shall initiate the release procedures appropriate to the class. The timer should be stopped when a DR TPDU or an N-DISCONNECT indication is received.

Note 5 – In classes other than 0, it is suggested that the invalid TPDU be also included in the ER TPDU.

6.23 *Splitting and recombining*

6.23.1 *Purpose*

This procedure is used only in Class 4 to allow a transport connection to make use of multiple network connections to provide additional resilience against network failure, to increase throughput, or for other reasons.

6.23.2 *Procedure*

When this function is being used, a transport connection may be assigned (see § 6.1) to multiple network connections (see Note 1). TPDUs for the connection may be sent over any such network connection.

If the use of Class 4 is not accepted by the remote transport entity following the negotiation rules, then no network connection except that over which the CR TPDU was sent may have this transport connection assigned to it.

Note 1 – The resequencing function of Class 4 (see § 6.20) is used to ensure that TPDUs are processed in the correct sequence.

Note 2 – Either transport entity may assign the connection to further network connections of which it is the owner at any time during the lifetime of the transport connection, provided the following constraints are respected:

- the initiator does not start splitting before having received the CC TPDU;
- as soon as a new assignment is done, it is recommended to send a TPDU on this network connection in order to make the remote entity aware of this assignment.

Note 3 – In order to enable the detection of unsignalled network connection failures, a transport entity performing splitting should ensure that TPDUs are sent at intervals on each supporting network connection, for example by sending successive TPDUs on successive network connections, where the set of network connections is used cyclically. By monitoring each network connection, a transport entity may detect unsignalled network connection failures, following the inactivity procedures defined in § 12.2.3.3. Thus, for each network connection, no period I (see § 12.2.3.1) may elapse without the receipt of some TPDU for some transport connection.

7 **Protocol classes**

Table 6/X.224 gives an overview of which elements of procedure are included in each class. In certain cases, the elements of procedure within different classes are not identical, and, for this reason, Table 6/X.224 cannot be considered part of the definitive specification of the protocol.

8 **Specification for Class 0: simple class**

8.1 *Functions of Class 0*

Class 0 is designed to have minimum functionality. It provides only the functions needed for connection establishment with negotiation, data transfer with segmenting and protocol error reporting.

Class 0 provides transport connections with flow control based on the network service provided flow control, and disconnection based on the network service disconnection.

8.2 *Procedures for Class 0*

8.2.1 *Procedures applicable at all times*

The transport entities shall use the following procedures:

a) TPDU transfer (see § 6.2);

b) association of TPDUs with transport connections (see § 6.9);

c) treatment of protocol errors (see § 6.22);

d) error release (see § 6.8).

TABLE 6/X.224

Allocation of elements of procedure within classes

Procedure	Cross ref.	Variant	0	1	2	3	4
Assignment to network conn.	6.1		X	X	X	X	X
TPDU Transfer	6.2		X	X	X	X	X
Segmenting and reassembling	6.3		X	X	X	X	X
Concatenation and separation	6.4			X	X	X	X
Connection Establishment	6.5		X	X	X	X	X
Connection Refusal	6.6		X	X	X	X	X
Normal Release	6.7	implicit explicit	X	 X	 X	 X	 X
Error Release	6.8		X		X		
Association of TPDUs with TCs	6.9		X	X	X	X	X
Data TPDU Numbering	6.10	normal extended		X	m[a)] o[a)]	m o	m o
Expedited Data Transfer	6.11	network normal network express		m ao	X[a)]	X	X
Reassignment after failure	6.12			X		X	[c)]
Retention until acknowledgement of TPDUs	6.13	Conf. receipt AK		ao m		 X	 X
Resynchronization	6.14			X		X	[c)]
Multiplexing and Demultiplexing	6.15				X[b)]	X	X
Explicit Flow Control (with) Explicit Flow Control (without)	6.16		 X	 X	m o	X	X
Checksum (use of) Checksum (non-use of)	6.17		 X	 X	 X	 X	X o
Frozen References	6.18			X		X	X
Retransmission on Timeout	6.19						X
Resequencing	6.20						X
Inactivity Control	6.21						X
Treatment of Protocol Errors	6.22		X	X	X	X	X
Splitting and Recombining	6.23						X

X: Procedure always included in class.
empty square: Not applicable.
m: Negotiable procedure whose implementation in equipment is mandatory.
o: Negotiable procedure whose implementation in equipment is optional.
ao: Negotiable procedure whose implementation in equipment is optional and where use depends on availability within the network service.

[a)] Not applicable in class 2 when non-use of explicit flow control is selected.

[b)] Multiplexing may lead to degradation of the quality of service if the non-use of explicit flow control has been selected.

[c)] This function is provided in class 4 using procedures other than those used in the cross reference.

8.2.2 *Connection establishment*

The transport entities shall use the following procedures:

a) assignment to network connection (see § 6.1); then

b) connection establishment (see § 6.5) and, if appropriate, connection refusal (see § 6.6);

subject to the following constraints:

c) the CR and CC TPDUs shall contain no parameter field other than those for TSAP-ID and maximum TPDU size;

d) the CR and CC TPDUs shall not contain a data field.

8.2.3 *Data transfer*

The transport entities shall use the segmenting and reassembling procedure (see § 6.3).

8.2.4 *Release*

The transport entities shall use the implicit variant of the normal release procedure (see § 6.7).

Note – The lifetime of the transport connection is directly correlated with the lifetime of the network connection.

9 **Specification for Class 1: basic error recovery class**

9.1 *Functions of Class 1*

Class 1 provides transport connections with flow control based on the network service provided flow control, error recovery, expedited data transfer, disconnection, and also the ability to support consecutive transport connections on a network connection.

This class provides the functionality of Class 0 plus the ability to recover after a failure signalled by the Network Layer, without involving the TS-user.

9.2 *Procedures for Class 1*

9.2.1 *Procedures applicable at all times*

The transport entities shall use the following procedures:

a) TPDU transfer (see § 6.2);

b) association of TPDU with transport connections (see § 6.9);

c) treatment of protocol errors (see § 6.22);

d) reassignment after failure (see § 6.12);

e) resynchronization (see § 6.14), or reassignment after failure (see § 6.12) together with resynchronization (see § 6.14);

f) concatenation and separation (see § 6.4);

g) retention until acknowledgment of TPDUs (see § 6.13); the variant used, AK or confirmation of receipt, shall be as selected during connection establishment (see Notes);

h) frozen references (see § 6.18).

Note 1 – The negotiation of the variant of retention until acknowledgment of TPDUs procedure to be used over the transport connection has been designed such that if the initiator proposes the use of the AK variant (i.e. the mandatory implementation option), the responder has to accept use of this option and if the initiator proposes use of the confirmation of receipt variant the responder is entitled to select use of the AK variant.

Note 2 – The AK variant makes use of AK TPDUs to release copies of retained DT TPDUs. The CDT parameter of AK TPDUs in Class 1 is not significant, and is set to 1111.

Note 3 – The confirmation of receipt variant is restricted to this class and its use depends on the availability of the network layer receipt confirmation service, and the expected cost reduction.

9.2.2 *Connection establishment*

The transport entities shall use the following procedures:

a) assignment to network connection (see § 6.1); then

b) connection establishment (see § 6.5) and, if appropriate, connection refusal (see § 6.6).

9.2.3 *Data transfer*

9.2.3.1 *General*

The sending transport entity shall use the following procedures:

a) segmenting (see § 6.3); then

b) the normal format variant of DT TPDU numbering (see § 6.10).

The receiving transport entity shall use the following procedures:

c) the normal format variant of DT TPDU numbering (see § 6.10); then

d) reassembling (see § 6.3).

Note 1 – The use of RJ TPDU during resynchronization (see § 6.14) can lead to retransmission. Thus, the receipt of a duplicate DT TPDU is possible; such a DT TPDU is discarded.

Note 2 – It is possible to decide on a local basis to issue an N-RESET request in order to force the remote entity to carry out the resynchronization (see § 6.14).

9.2.3.2 *Expedited data*

The transport entities shall use either of the network normal data or the network expedited variants of the expedited data transfer procedure (see § 6.11) if their use has been selected during connection establishment (see Note 1).

The sending transport entity shall not allocate the same ED-TPDU-NR to successive ED TPDUs (see Notes 2 and 3).

When acknowledging an ED TPDU by sending an EA TPDU the transport entity shall put into the YR-EDTU-NR parameter of the EA TPDU the value received in the ED-TPDU-NR parameter of the ED TPDU.

Note 1 – The negotiation of the variant of expedited data transfer procedure to be used over the transport connection has been designed such that if the initiator proposes the use of the network normal data variant (i.e., the mandatory implementation option), the responder has to accept use of this option and if the initiator proposes use of the network expedited variant, the responder is entitled to select use of the network normal data variant.

Note 2 – This numbering enables the receiving transport entity to discard repeated ED TPDUs when resynchronization (see § 6.14) has taken place.

Note 3 – No other significance is attached to the ED-TPDU-NR parameter. It is suggested, but not essential, that the values used be consecutive modulo 128.

9.2.4 *Release*

The transport entities shall use the explicit variant of the release procedure (see § 6.7).

10 Specification for Class 2: multiplexing class

10.1 *Functions of Class 2*

Class 2 provides transport connections with or without individual flow control; no error detection or error recovery is provided.

If the network connection resets or disconnects, the transport connection is terminated without the transport release procedure and the TS-user is informed.

When explicit flow control is used, a credit mechanism is defined allowing the receiver to inform the sender of the exact amount of data he is willing to receive and expedited data transfer is available.

10.2 *Procedures for Class 2*

10.2.1 *Procedures applicable at all times*

The transport entities shall use the following procedures:

a) association of TPDUs with transport connections (see § 6.9);

b) TPDU transfer (see § 6.2);

c) treatment of protocol errors (see § 6.22);

d) concatenation and separation (see § 6.4);

e) error release (see § 6.8).

Additionally the transport entities may use the following procedure:

f) multiplexing and demultiplexing (see § 6.15).

10.2.2 *Connection establishment*

The transport entities shall use the following procedures:

a) assignment to network connection (see § 6.1); then

b) connection establishment (see § 6.5) and, if applicable, connection refusal (see § 6.6).

10.2.3 *Data transfer when non-use of explicit flow control has been selected*

If this option has been selected as a result of the connection establishment, the transport entities shall use the segmenting procedure (see § 6.3).

The TPDU-NR field of DT TPDUs is not significant and may take any value.

Note – Expedited data transfer is not applicable (see § 6.5).

10.2.4 *Data transfer when use of explicit flow control has been selected*

10.2.4.1 *General*

The sending transport entity shall use the following procedures:

a) segmenting (see § 6.3); then

b) DT TPDU numbering (see § 6.10);

The receiving transport entity shall use the following procedures:

c) DT TPDU numbering (see § 6.10); if a DT TPDU is received which is out of sequence it shall be treated as a protocol error; then

d) reassembling (see § 6.3).

The variant of the DT TPDU numbering which is used by both transport entities shall be that which was agreed at connection establishment.

10.2.4.2 *Flow control*

The transport entities shall send an initial credit (which may be zero) in the CDT field of the CR or CC TPDU. This credit represents the initial value of the upper window allocated to the peer entity.

The transport entity that receives the CR or the CC TPDU shall consider its lower window edge as zero, and its upper window edge as the value of the CDT field in the received TPDU.

In order to authorize the transmission of DT TPDUs by its peer, a transport entity may transmit an AK TPDU at any time, subject to the following constraints:

a) the YR-TU-NR parameter shall be at most one greater than the TPDU-NR parameter of the last received DT TPDU or shall be zero if no DT TPDU has been received;

b) if an AK TPDU has previously been sent, the value of the YR-TU-NR parameter shall not be lower than that in the previously sent AK TPDU;

c) the sum of the YR-TU-NR and CDT parameters shall not be less than the upper window edge allocated to the remote entity (see Note 1).

A transport entity which receives an AK TPDU shall consider the YR-TU-NR parameter as its new lower window edge, and the sum of YR-TU-NR and CDT as its new upper window edge. If either of these have been reduced or if the lower window edge has become more than one greater than the TPDU-NR of the last transmitted DT TPDU, this shall be treated as a protocol error (see § 6.22).

A transport entity shall not send a DT TPDU with a TPDU-NR outside of the transmit window (see Notes 2 and 3).

Note 1 – This means that credit reduction is not applicable.

Note 2 – This means that a transport entity is required to stop sending if the TPDU-NR parameter of the next DT TPDU which would be sent would be the upper window edge. Sending of DT TPDU may be resumed if an AK TPDU is received which increases the upper window edge.

Note 3 – The rate at which a transport entity progresses the upper window edge allocated to its peer entity constrains the throughput attainable on the transport connection.

10.2.4.3 *Expedited data*

The transport entities shall follow the network normal data variant of the expedited data transfer procedure in § 6.11 if its use has been agreed during connection establishment. ED and EA TPDUs are not subject to the flow control procedures in § 10.2.4.2. The ED-TPDU-NR and YR-EDTU-NR parameters of ED and EA TPDUs respectively are not significant and may take any value.

10.2.5 *Release*

The transport entities shall use the explicit variant of the release procedure in § 6.7.

11 Specification for Class 3: error recovery and multiplexing class

11.1 *Functions of Class 3*

This class provides the functionality of Class 2 (with use of explicit flow control) plus the ability to recover after a failure signalled by the Network Layer without involving the TS-user.

The mechanisms used to achieve this functionality also allow the implementation of more flexible flow control.

11.2 *Procedures for Class 3*

11.2.1 *Procedures applicable at all times*

The transport entities shall use the following procedures:

a) association of TPDUs with transport connections (see § 6.9);

b) TPDU transfer (see § 6.2) and retention until acknowledgment of TPDUs (AK variant only) (see § 6.13);

c) treatment of protocol errors (see § 6.22);

d) concatenation and separation (see § 6.4);

e) reassignment after failure (see § 6.12), together with resynchronization (see § 6.14);

f) frozen references (see § 6.18).

Additionally, the transport entities may use the following procedure:

g) multiplexing and demultiplexing (see § 6.15).

11.2.2 *Connection establishment*

The transport entity shall use the following procedures:

a) assignment to network connections (see § 6.1); then

b) connection establishment (see § 6.5) and, if appropriate, connection refusal (see § 6.6).

11.2.3 *Data transfer*

11.2.3.1 *General*

The sending transport entity shall use the following procedures:

a) segmenting (see § 6.3); then

b) DT TPDU numbering (see § 6.10); after receipt of an RJ TPDU (see § 11.2.3.2) the next DT TPDU to be sent may have a value which is not the previous value of TPDU-NR plus one.

The receiving transport entity shall use the following procedures:

c) DT TPDU numbering (see § 6.10); the TPDU-NR parameter of each received DT TPDU shall be treated as a protocol error if it exceeds the greatest such value received in a previous DT TPDU by more than one (see Note); then

d) reassembling (see § 6.3); duplicated TPDUs shall be eliminated before reassembling is performed.

Note – The use of RJ TPDUs (see § 11.2.3.2) can lead to retransmission and reduction of credit. Thus the receipt of a DT TPDU which is a duplicate, or which is greater than or equal to the upper window edge allocated to the peer entity, is possible and is therefore not treated as a protocol error.

11.2.3.2 *Use of RJ TPDU*

A transport entity may send an RJ TPDU at any time in order to invite retransmission or to reduce the upper window edge allocated to the peer entity (see Note 1).

When an RJ TPDU is sent, the following constraints shall be respected:

a) the YR-TU-NR parameter shall be at most one greater than the greatest such value received in a previous DT TPDU, or shall be zero if no DT TPDU has yet been received (see Note 2);

b) if an AK or RJ TPDU has previously been sent, the YR-TU-NR parameter shall not be lower than that in the previously sent AK or RJ TPDU.

When a transport entity receives an RJ TPDU (see Note 3):

c) the next DT TPDU to be transmitted, or retransmitted, shall be that for which the value of the TPDU-NR parameter is equal to the value of the YR-TU-NR parameter of the RJ TPDU;

d) the sum of the values of the YR-TU-NR and CDT parameters of the RJ TPDU becomes the new upper window edge (see Note 4).

Note 1 – An RJ TPDU can also be sent as part of the resynchronization (see § 6.14) and reassignment after failure (see § 6.12) procedures.

Note 2 – It is suggested that the YR-TU-NR parameter be equal to the TPDU-NR parameter of the next expected DT TPDU.

Note 3 – These rules are a subset of those specified for when an RJ TPDU is received during resynchronization (see § 6.14) and reassignment after failure (see § 6.12).

Note 4 – This means that RJ TPDU can be used to reduce the upper window edge allocated to the peer entity (credit reduction).

11.2.3.3 *Flow control*

The procedures shall be as defined in § 10.2.4.2, except that:

a) a credit reduction may lead to the reception of a DT TPDU with a TPDU-NR parameter whose value is not but would have been less than the upper window edge allocated to the remote entity prior to the credit reduction. This shall not be treated as a protocol error;

b) receipt of an AK TPDU which sets the lower window edge more than one greater than the TPDU-NR of the last transmitted DT TPDU shall not be treated as a protocol error, provided that all acknowledged DT TPDUs have been previously transmitted (see Notes 1 and 2).

Note 1 – This can only occur during retransmission following receipt of an RJ TPDU.

Note 2 – The transport entity may either continue retransmission as before or retransmit only those DT TPDUs not acknowledged by the AK TPDU. In either case, copies of the acknowledged DT TPDUs need not be retained further.

11.2.3.4 *Expedited data*

The transport entities shall follow the network normal data variant of expedited data transfer procedure in § 6.11 if its use has been agreed during connection establishment.

The sending transport entity shall not allocate the same ED-TPDU-NR to successive ED TPDUs.

The receiving transport entity shall transmit an EA TPDU with the same value in its YR-EDTU-NR-parameter. If, and only if, this number is different from that of the previously received ED TPDU, shall it generate a T-EXPEDITED DATA indication to convey the data to the TS-user (see Note 2).

Note 1 – No other significance is attached to the ED-TPDU-NR parameter. It is suggested, but not essential, that the values be consecutive modulo 2^n, where n is the number of bits of the parameter.

Note 2 – This procedure ensures that the TS-user does not receive data corresponding to the same ED TPDU more than once.

11.2.4 *Release*

The transport entities shall use the explicit variant of the release procedure (see § 6.7).

12 Specification for Class 4: error detection and recovery class

12.1 *Functions of Class 4*

Class 4 provides the functionality of Class 3, plus the ability to detect and recover from lost, duplicated or out of sequence TPDUs without involving the TS-user.

Class 4 detects signalled and unsignalled network failures (i.e. resets or disconnects or inactivity) and recovers from these failures by using timeout mechanisms.

This detection of errors is made by extended user of the sequence numbering of Classes 2 and 3, by timeout mechanisms, and by additional procedures.

This class additionally detects and recovers from damaged TPDUs by using a checksum mechanism. The use of the checksum mechanism must be available but its use or its non-use is subject to negotiation. Furthermore, this class provides additional resilience against network failure and increased throughput capability by allowing a transport connection to make use of multiple network connections.

12.2 *Procedures for Class 4*

12.2.1 *Procedures available at all times*

12.2.1.1 *Timers used at all times*

This sub-clause defines timers that apply at all times in Class 4. These timers are listed in Table 7/X.224.

TABLE 7/X.224

Timer parameters related to the operation of Class 4

Symbol	Name	Definition
M_{LR}	NSDU lifetime local-to-remote	A time bound for the maximum time which may elapse between the transmission of an NSDU by a local transport entity and the receipt of any copy of it by a remote peer entity.
M_{RL}	NSDU lifetime remote-to-local	A time bound for the maximum time which may elapse between the transmission of an NSDU by a remote transport entity and the receipt of any copy of it by a local peer entity.
E_{LR}	Expected maximum transit delay local-to-remote	A time bound for the maximum delay suffered by all but a small proportion of NSDUs transferred from the local transport entity to a remote peer entity.
E_{LR}	Expected maximum transit delay remote-to-local	A time bound for the maximum delay suffered by all but a small proportion of NSDUs transferred from a remote transport entity to the local peer entity.
A_L	Local acknowledge time	A time bound for the maximum time which can elapse between the receipt of a TPDU by the local transport entity from the network layer and the transmission of the corresponding acknowledgement.
A_R	Remote acknowledge time	As A_L, but for the remote entity.
T1	Local retransmission time	A time bound for the maximum time the local transport entity will wait for acknowledgement before retransmitting a TPDU.
R	Persistence time	A time bound for the maximum time that the local transport entity will continue to transmit a TPDU that requires acknowledgement.
N	Maximum number of transmissions	A bound for the number of times which the local transport entity will continue to transmit a TPDU that requires acknowledgement.
L	Time bound on references and sequence numbers	A time bound for the maximum time between the transmission of a TPDU and the receipt of any acknowledgement relating to it.
I	Inactivity time	A time bound for the time after which a transport entity will, if it does not receive a TPDU, initiate the release procedure to terminate the transport connection. *Note* – This parameter is required for protection against unsignalled failures.
W	Window time	A time bound for the maximum time a transport entity will wait before retransmitting up-to-date window information.

This Recommendation does not define specific values for the timers, and the derivations described in this subclause are not mandatory. The values should be chosen so that the required quality of service can be provided, given the known characteristics of the network.

Timers that apply only to specific procedures are defined under the appropriate procedure.

12.2.1.1.1 *NSDU lifetimes (M_{LR} , M_{RL})*

The network layer is assumed to provide, as an aspect of its quality of service, for a bound on the maximum lifetime of NSDUs in the network. This value may be different in each direction of transfer through a network between two transport entities. The values, for both directions of transfer, are assumed to be known by the transport entities.

The maximum NSDU lifetime local-to-remote (M_{LR}) is the maximum time which may elapse between the transmission of an NSDU from the local transport entity to the network layer and the receipt of any copy of the NSDU from the network layer at the remote transport entity.

The maximum NSDU lifetime remote-to-local (M_{RL}) is the maximum time which may elapse between the transmission of an NSDU from the remote transport entity to the network layer and receipt of any copy of the NDSU from the network layer at the local transport entity.

12.2.1.1.2 *Expected maximum transit delay (E_{LR} ,E_{RL})*

The network layer is assumed to provide, as an aspect of its quality of service, an expected maximum transit delay for NSDUs in the network. This value may be different in each direction of transfer through a network between two transport entities. The values, for both directions of transfer, are assumed to be known by the transport entities.

The expected maximum transit delay local-to-remote (E_{RL}) is the maximum delay suffered by all but a small proportion of NSDUs transferred through the network from the local transport entity to the remote transport entity.

The expected maximum transit delay remote-to-local (E_{RL}) is the maximum delay suffered by all but a small proportion of NSDUs transferred through the network from the remote transport entity to the local transport entity.

12.2.1.1.3 *Acknowledge time (A_R , A_L)*

Any transport entity is assumed to provide a bound for the maximum time which can elapse between its receipt of a TPDU from the Network Layer and its transmission of the corresponding response. This value is referred to as A_L. The corresponding time given by the remote transport entity is referred to as A_R.

12.2.1.1.4 *Local retransmission time (T1)*

The local transport entity is assumed to maintain a bound on the time it will wait for an acknowledgment before retransmitting the TPDU.

Its value is given by:

$$T1 = E_{LR} + E_{RL} + A_R + X$$

where:

E_{LR} = expected maximum transit delay local-to-remote,

E_{RL} = expected maximum transit delay remote-to-local,

A_R = remote acknowledge time, and

X = local processing time for a TPDU.

Note – During connection establishment the value of A_R is not known. In this case a suitable bound for T1 may be established either by estimating (or having "a priori" knowledge of) A_R or by applying a suitable algorithm to the TC establishment delay QOS parameter.

12.2.1.1.5 *Persistence time (R)*

The local transport entity is assumed to provide a bound for the maximum time for which it may continue to retransmit a TPDU requiring positive acknowledgement and which is not outside the current transmit window, even after credit reduction. This value is referred to as R.

This value is clearly related to the time elapsed between retransmission, T1, and the maximum number of retransmissions, N. It is not less than T1 x (N – 1) + x, where x is a small quantity to allow for additional internal delays, the granularity of the mechanism used to implement T1 and so on. Because R is a bound, the exact value of x is unimportant as long as it is bounded and the value of a bound is known.

12.2.1.1.6 *Time bound on references and sequence numbers (L)*

A time bound for the maximum time between the decision to transmit a TPDU and the receipt of any acknowledgement relating to it (L) is given by:

$$L = M_{LR} + M_{RL} + R + A_R$$

where:

M_{LR} = NSDU lifetime local-to-remote,

M_{RL} = NSDU lifetime remote-to-local,

A_R = remote acknowledge time, and

R = persistence time.

It is necessary to wait for a period L before re-using any reference or sequence number, to avoid confusion in case a TPDU referring to it may be duplicated or delayed.

Note 1 – In practice, the value of L may be unacceptably large. It may also be only a statistical figure at a certain confidence level. A smaller value may therefore be used where this still allows the required quality of service to be provided.

Note 2 – The relationships between the times discussed above are illustrated in Figures 3 and 4/X.224.

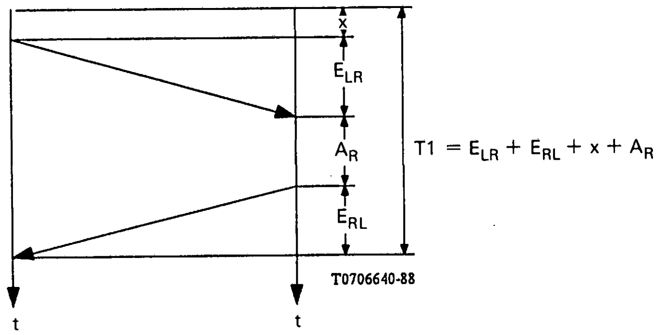


FIGURE 3/X.224

The interrelationship of times for the average case in class 4

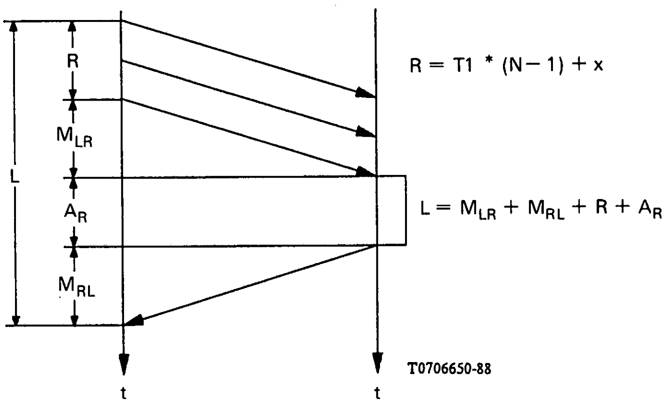


FIGURE 4/X.224

The interrelationship of times for the maximum delay in class 4

12.2.1.2 *General procedures*

The transport entity shall use the following procedures:

a) TPDU transfer (see § 6.2);

b) association of TPDUs with transport connections (see § 6.9);

c) treatment of protocol errors (see § 6.22);

d) checksum (see § 6.17);

e) splitting and recombining (see § 6.23);

f) multiplexing and demultiplexing (see § 6.15);

g) retention until acknowledgement of TPDUs (see § 6.13);

h) frozen reference (see § 6.18);

i) retransmission procedures; when a transport entity has some outstanding TPDUs that require acknowledgement, it will check that no T1 interval elapses without the arrival of a TPDU that acknowledges at least one of the outstanding TPDUs.

If the timer expires, except if the TPDU to be retransmitted is a DT TPDU and it is outside the transmit window due to credit reduction, the first TPDU is retransmitted and the timer is restarted. After N transmissions (i.e. N-1 retransmissions), it is assumed that useful two-way communication is no longer possible and the release procedure is used, and the TS-user is informed.

Note 1 – This procedure may be implemented by different means. For example:

a) one interval is associated with each TPDU. If the timer expires, the associated TPDU will be retransmitted, and the timer T1 will be restarted for all subsequent DT TPDUs; or

b) one interval is associated with each transport connection:

   1) if the transport entity transmits a TPDU requiring acknowledgement, it starts timer T1;

   2) if the transport entity receives a TPDU that acknowledges one of the TPDUs to be acknowledged, it restarts timer T1 unless the received TPDU is an AK TPDU which explicitly closes the transmit window;

   3) if the transport entity receives a TPDU that acknowledges the last TPDU to be acknowledged, it stops timer T1.

For a decision whether the retransmission timer T1 is maintained on a per TPDU or on a per transport connection basis, throughput considerations have to be taken into account.

Note 2 – For DT TPDUs, it is a local choice to retransmit either only the first DT TPDU or all TPDUs waiting for an acknowledgement up to the upper window edge.

Note 3 – It is suggested that after N transmissions of a DT TPDU, the transport entity waits T1 + W + M_{RL} to provide a higher possibility of receiving an acknowledgement before entering the release phase.

For other TPDU types which may be retransmitted, it is suggested that after N transmissions, the transport entity waits T1 + M_{RL} to provide a higher possibility of receiving the expected reply.

12.2.2 *Procedures for connection establishment*

12.2.2.1 *Timers used in connection establishment*

There are no timers specific to connection establishment.

12.2.2.2 *General procedures*

The transport entities shall use the following procedures:

a) when a network connection to which the transport connection is assigned is released (N DISind received):

   1) if a CC TPDU is awaited the initiator shall perform a new assignment according to QOS and retransmission procedure (i.e., no more than N × T1 keeping sending CR TPDU);

2) if there is at least one other network connection to which the tranport connection is assigned both initiator and acceptor may either perform a new assignment or continue operation using one of the remaining network connections;

3) if the transport connection becomes unassigned the acceptor may either perform new assignment or wait (there is no risk of deadlock since either T1 or I is running), the initiator shall perform a new assignment (except in the closing state);

b) connection establishment (see § 6.5) and, if appropriate, connection refusal (see § 6.6) together with the additional procedures:

1) a connection is not considered established until the successful completion of a 3-way TPDU exchange. The sender of a CR TPDU must respond to the corresponding CC TPDU by immediately sending a DT, ED, DR or AK TPDU;

2) as a result of duplication or retransmission, a CR TPDU may be received specifying a source referene which is already in use with the sending transport entity. If the receiving transport entity is in the data transfer phase, having completed the 3-way TPDU exchange procedure, or is waiting for the T-CONNECT response from the TS-user, the receiving transport entity shall discard such a TPDU. Otherwise, a CC TPDU shall be transmitted;

3) as a result of duplication or retransmission, a CC TPDU may be received specifying a paired reference which is already in use. The receiving transport entity shall only acknowledge the duplicate CC TPDU according to the procedure in § 12.2.2.2, b) 1);

4) a CC TPDU may be received specifying a reference which is in the frozen state. The response to such a TPDU shall be a DR TPDU;

5) the retransmission procedures (see § 12.2.1.2) are used for both the CR TPDU and CC TPDU.

Note – After receiving a CR TPDU, it is recomended that the transport entity enforce a time limit upon the Transport Service user so that its late acceptance of the transport connection will not cause a delayed CC TPDU to be sent.

12.2.3 *Procedures for data transfer*

12.2.3.1 *Timers used in data transfer*

The data transfer procedures use two additional timers.

12.2.3.1.1 *Inactivity time (I)*

To protect against unsignalled breaks in the network connection, or failure of the peer transport entity, (half-open connections), each transport entity maintains an inactivity time interval.

Note – A suitable value for I is given by 2 × [N × maximum of (T1, W)] unless local needs indicate another more appropriate value.

12.2.3.1.2 *Window time (W)*

A transport entity maintains a timer interval to ensure that there is a bound on the maximum interval between window updates.

12.2.3.2 *General procedures for data transfer*

The transport entities shall use the following procedures:

a) inactivity control (see § 6.21);

b) expedited data (see § 6.11);

c) explicit flow control (see § 6.16).

The sending transport entity shall use the following procedures in the following order:

d) segmenting (see § 6.3);

e) DT TPDU numbering (see § 6.10).

The receiving transport entity shall use the following procedures in the following order:

f) DT TPDU numbering (see § 6.10);

g) resequencing (see § 6.20);

h) reassembling (see § 6.3).

12.2.3.3 *Inactivity control*

If the interval of the inactivity timer I expires without receipt of some TPDU, the transport entity shall initiate the release procedures. To prevent expiration of the remote transport entity's inactivity timer when no data is being sent, the local transport entity must send AK TPDUs at suitable intervals in the absence of data, having regard to the probability of TPDU loss. The window synchronization procedures (see § 12.2.3.8) ensure that this requirement is met.

Note – It is likely that the release procedures initiated due to inactivity timer expiration will fail, as such expiration indicates probable failure of the supporting network connection or of the remote transport entity.

12.2.3.4 *Expedited data*

The transport entities shall follow the network normal data variant of the expedited data transfer procedures (see § 6.11), if the use of transport expedited data transfer service option has been agreed during connection establishment.

The ED TPDU shall have a TPDU-NR which is allocated from a separate sequence space from that of the DT TPDUs. A transport entity shall allocate the sequence number zero to the ET-TPDU-NR of the first ED TPDU which it transmits for a transport connection. For subsequent ED TPDUs sent on the same transport connection, the transport entity shall allocate a sequence number one greater than the previous one.

Modulo 2^7 arithmetic shall be used when normal formats have been selected and modulo 2^{31} arithmetic shall be used when extended formats have been selected.

The receiving transport entity shall transmit an EA TPDU with the same sequence number in its YR-EDTU-NR parameter. If this number is one greater than in the previously received in-sequence ED TPDU, the receiving transport entity shall transfer the data in the ED TPDU to the TS-user.

If a transport entity does not receive an EA TPDU in acknowledgement to an ET TPDU, it shall follow the retransmission procedures (see Note and § 12.2.1.2).

The sender of an ED-TPDU shall not send any new DT TPDUs created from a T-DATA request subsequent to the T-EXPEDITED DATA request, until it receives the EA TPDU.

Note – This procedure ensures that ED TPDUs are delivered to the TS-user in sequence and that the TS-user does not receive data corresponding to the same ED TPDU more than once. Also, it guarantees the arrival of the ED TPDU before any data subsequently sent by the TS-user.

12.2.3.5 *Resequencing*

The receiving transport entity shall deliver all DT TPDUs to the TS-user in the order specified by the TPDU-NR parameter.

DT TPDUs received out of sequence but within the transmit window shall not be delivered to the TS-user until in-sequence TPDUs have also been received. DT TPDUs received out-of-sequence and outside the transmit window shall be discarded, but may result in transmission of an AK TPDU with up-to-date window information (see § 12.2.3.8).

Duplicate TPDUs can be detected because the sequence number matches that of previously received TPDUs. Sequence numbers shall not be reused for the period L after their previous use. Otherwise, a new, valid TPDU could be confused with a duplicated TPDU which had previously been received and acknowledged.

Duplicated DT TPDUs shall be acknowledged, since the duplicated TPDU may be the result of a retransmmission resulting from the loss of an AK TPDU.

The data contained in a duplicated DT TPDU shall be discarded.

12.2.3.6 *Explicit flow control*

The transport entities shall send an initial credit (which may take the value 0) in the CDT parameter of the CR TPDU or CC TPDU. This credit represents the initial value of the upper window edge of the peer entity.

The transport entity which receives the CR TPDU or CC TPDU shall consider its lower window edge as zero and its upper window edge as the value in the CDT parameter in the received TPDU.

In order to authorize the transmission of DT TPDUs by its peer, a transport entity may transmit an AK TPDU at any time.

The sequence number of an AK TPDU shall not exceed the sequence number of the next expected DT TPDU, i.e., shall not be greater than the highest sequence number of a received DT TPDU, plus one.

A transport entity may send a duplicate AK TPDU containing the same sequence number, CDT, and subsequence number field at any time.

A transport entity may increase or decrease the upper window edge at any time.

A transport entity which receives an AK TPDU shall consider the value of the YR-TU-NR parameter as its new lower window edge if it is greater than any previously received in a YR-TU-NR parameter, and the sum of YR-TU-NR and CDT as its new upper window edge subject to the procedures for sequencing AK TPDUs (see § 12.2.3.8). A transport entity shall not transmit or retransmit a DT TPDU with a sequence number outside the transmit window.

12.2.3.7 *Sequencing of received AK TPDUs*

To allow a receiving transport entity to properly sequence a series of AK TPDUs that all contain the same sequence number and thereby use the correct CDT value, AK TPDUs may contain a sub-sequence parameter. For the purpose of determining the correct sequence of AK TPDUs, the absence of the sub-sequence parameter shall be equivalent to the value of the parameter set to zero.

An AK TPDU is defined to be in sequence if:

a) the sequence number is greater than in any previously received AK TPDU, or

b) the sequence number is equal to the highest in any previously received AK TPDU, and the sub-sequence parameter is greater than in any previously received AK TPDU having the same value of YR-TU-NR field, or

c) the sequence number and sub-sequence parameter are both equal to the highest in any previously received AK TPDU, and the CDT parameter is greater than or equal to that in any previously received AK TPDU having the same YR-TU-NR parameter.

When the receiving transport entity recognizes an out of sequence AK TPDU, it shall discard it.

12.2.3.8 *Procedures for transmission of AK TPDUs*

12.2.3.8.1 *Transmission of AK TPDUs*

An in-sequence DT TPDU shall be acknowledged within time A_L, by the transmission of AK TPDU whose "YR-TU-NR" field is set to at least the sequence number of the received DT TPDU plus one.

An AK TPDU shall be transmitted containing up-to-date window information if:

a) a DT TPDU is received whose sequence number is lower than the lower window edge, but greater than or equal to the lower window edge minus the maximum credit value ever given for this transport connection, or

b) a DT TPDU is received whose sequence number is above the current upper window but following credit reduction is within the upper window edge which has been granted and then withdrawn.

Note 1 – A simpler implementation may send an AK TPDU upon receipt of any DT TPDU outside the transmit window.

Note 2 – The procedure a) is required so that loss of an AK TPDU is correctly recovered, i.e., when the sender of the DT TPDU retransmits it following nonreceipt of an acknowledgement.

Note 3 – The procedure b) is required due to the possibility of loss of the AK TPDU indicating the upper window edge reduction, which could otherwise cause incorrect termination of the transport connection.

A transport entity shall not allow an interval W to pass without the transmission of an AK TPDU. If the transport entity is not using the procedure following setting CDT to zero (see § 12.2.3.8.3) or reduction of the upper window edge (see § 12.2.3.8.4), and does not have to acknowledge receipt of any DT TPDU, then it shall achieve this by retransmission of the most recent AK TPDU, with up-to-date window information.

Note – The use of the procedures defined in §§ 12.2.3.8.3 and 12.2.3.8.4 are optional for any transport entity. The protocol operates correctly either with or without these procedures which are defined to enhance the efficiency of its operation.

12.2.3.8.2 *Sequence control for transmission of AK TPDUs*

To allow the receiving transport entity to process AK TPDUs in the correct sequence, as described in § 12.2.3.7, the sub-sequence parameter shall be included following reduction of CDT. If the value of the sub-sequence number to be transmitted is zero, then the parameter should be omitted.

The value of the sub-sequence parameter, if used, shall be zero (either explicitly or by absence of the parameter) if the sequence number is greater than the parameter in previous AK TPDUs sent by the transport entity.

If the sequence number is the same as the previous AK TPDU sent and the CDT parameter is equal to or greater than the CDT parameter in the previous AK TPDU sent, then the sub-sequence parameter, if used, shall be equal to that in the previously sent AK TPDU.

If the sequence number is the same as the previous AK TPDU sent and the CDT parameter is less than the value of the CDT parameter in the previous AK TPDU sent, then the sub-sequence parameter, if used, shall be one greater than the value in the previous AK TPDU.

Note – If a transport entity never reduces credit then it does not need to use the sub-sequence parameter.

12.2.3.8.3 *Retransmission of AK TPDUs after CDT set to zero*

Due to the possibility of loss of AK TPDUs, the upper window edge as perceived by the transport entity transmitting an AK TPDU may differ from that perceived by the intended recipient. To avoid the possibility of extra delay, the retransmission procedure (see § 12.2.1.2) should be followed for an AK TPDU, if it opens the transmit window which has previously been closed by sending an AK TPDU with CDT field set to zero.

The retransmission procedure, if used, terminates and the procedure in § 12.2.3.8.1 is used when:

a) AK TPDU is received containing the flow control confirmation parameter, whose lower window edge and sub-sequence fields are equal to the sequence number and sub-sequence number in the retained AK TPDU and whose credit field is not zero;

b) an AK TPDU is transmitted with a sequence number higher than that in the retained AK TPDU, due to reception of a DT TPDU whose sequence number is equal to the lower window edge;

c) N transmissions of the retained AK TPDU have taken place. In this case, the transport entity shall continue to transmit the AK TPDU at an interval of W.

An AK TPDU which is subject to the retransmission procedure shall not contain the flow control confirmation parameter. If it is required to transmit this parameter concurrently, an additional AK TPDU shall be transmitted having the same values in the YR-TU-NR, sub-sequence (if applicable) and CDT parameters.

12.2.3.8.4 *Retransmission procedures following reduction of the upper window edge*

This subsection specifies the procedure for retransmission of AK TPDUs after a transport entity has reduced the upper window edge (see § 12.2.3.6) or for an AK TPDU with the credit field set to zero. This procedure is used until the lower window edge exceeds the highest value of the upper window edge ever transmitted (i.e. the value existing at the time of credit reduction, unless a higher value is retained from a previous credit reduction).

The retransmission procedure should be followed for any AK TPDU which increases the upper window edge, unless it is known that the remote transport entity has an open window. This is known if:

– a flow control confirmation (FCC) parameter has been received, corresponding to an AK TPDU transmitted following the most recent credit reduction, and

– this FCC parameter conveys an upper window edge value (i.e., the sum of the lower window edge and credit fields) which is greater than the lower window edge of the transmitted AK TPDU.

This retransmission procedure for any particular AK TPDU shall terminate when:

a) an AK TPDU is received containing the flow control confirmation parameter, whose lower window edge and sub-sequence fields are equal to the lower window edge (YR-TU-NR) and sub-sequence number in the retained AK TPDU; or

b) N transmission of the retained AK TPDU have taken place. In this case, the transport entity shall continue to transmit the AK TPDU as an interval of W.

An AK TPDU which is subject to the retransmission procedure shall not contain the flow control confirmation parameter. If it is required to transmit this parameter concurrently, an additional AK TPDU shall be transmitted having the same values in the sequence, sub-sequence (if applicable) and credit fields.

Note – Retransmission of AK TPDUs is normally not necessary, except following explicit closing of the window (i.e., transmission of an AK TPDU with CDT parameter set to zero). If data is available to be transmitted, the retransmission procedure for DT TPDUs will ensure that an AK TPDU is received granting further credit where this is available. Following credit reduction, this may no longer be so, because retransmission may be inhibited by the credit reduction. The rules described in this clause avoid extra delay.

The rules for determining whether to apply the retransmission procedure to an AK TPDU may be expressed alternatively as follows:

LWE = lower window edge

UWE = upper window edge

KUWE = lower bound on upper window edge held by remote transport entity.

The retransmission procedure is used whenever:

$$(\text{UWE} > \text{LWE}) \text{ and } (\text{KUWE} = \text{LWE})$$

i.e. when the window is opened and it is not known definitely that the remote transport entity is aware of this.

KUWE is maintained as follows. When credit is reduced, KUWE is set to LWE. Subsequently, it is increased only upon receipt of a valid flow control confirmation (i.e. one which matches the retained lower window edge and sub-sequence). In the case, KUWE is set to the implied upper window edge of the flow control confirmation, i.e. the sum of its lower window edge and credit fields. By this means, it can be ensured that KUWE is always less than or equal to the actual upper window edge in use by the transmitter of DT TPDUs.

12.2.3.9 *Use of flow control confirmation parameter*

At any time, an AK TPDU may be transmitted containing a flow control confirmation parameter. The lower window edge, the sub-sequence and the credit fields shall be set to the same values as the corresponding parameters in the most recently received in-sequence AK TPDU.

An AK TPDU containing a flow control confirmation parameter should be transmitted whenever:

a) a duplicate AK TPDU is received, with the value of YR-TU-NR, CDT, and sub-sequence fields equal to the most recently received AK TPDU, but not itself containing the flow control confirmation parameter;

b) an AK TPDU is received which increases the upper window edge but not the lower window edge, and the upper window edge was formerly equal to the lower edge; or

c) an AK TPDU is received which increases the upper window edge but not the lower window edge, and the lower window edge is lower than the highest value of the upper window edge ever received and subsequently reduced (i.e. following credit reduction).

12.2.4 *Procedures for release*

12.2.4.1 *Timers used for release*

There are no timers used only for release.

12.2.4.2 *General procedures for release*

The transport entity shall use the explicit variant of normal release (see § 6.7).

13 Structure and encoding of TPDUs

13.1 *Validity*

Table 8/X.224 specifies those TPDUs which are valid for each class and the code for each TPDU.

13.2 *Structure*

All the transport protocol data units (TPDUs) shall contain an integral number of octets. The octets in a TPDU are numbered starting from 1 and increasing in the order they are put into an NSDU. The bits in an octet are numbered from 1 to 8, where bit 1 is the low-order bit.

When consecutive octets are used to represent a binary number or a binary coded decimal number (one digit per octet), the lower octet number has the most significant value.

Note 1 – The numbering of bits within an octet is a convention local to this Recommendation.

Note 2 – The use of the terms "high order" and "low order" is common to this Recommendation and to adjacent layer Recommendations.

Note 3 – The use of the above conventions does not affect the order of the bit transmission on a serial communications link.

Note 4 – As described in § 6.2.3, both transport entities respect these bit and octet ordering conventions, thus allowing communication to take place.

TABLE 8/X.224

TPDU codes

	Validity within classes					See	Code
	0	1	2	3	4		
CR: Connection Request	x	x	x	x	x	§ 13.3	1110 xxxx
CC: Connection Confirm	x	x	x	x	x	§ 13.4	1101 xxxx
DR: Disconnect Request	x	x	x	x	x	§ 13.5	1000 0000
DC: Disconnect Confirm		x	x	x	x	§ 13.6	1100 0000
DT: Data	x	x	x	x	x	§ 13.7	1111 0000
ED: Expedited Data		x	NF	x	x	§ 13.8	0001 0000
AK: Data Acknowledgement		NRC	NF	x	x	§ 13.9	0110 zzzz
EA: Expedited Data Acknowledgement		x	NF	x	x	§ 13.10	0010 0000
RJ: Reject		x		x		§ 13.11	0101 zzzz
ER: TPDU Error	x	x	x	x	x	§ 13.12	0111 0000
PI: Transport Protocol Id.						Annex B	0000 0001
Not available (see Note)						–	0000 0000
						–	0011 0000
						–	1001 xxxx
						–	1010 xxxx

xxxx (bits 4 to 1): used to signal the CDT in classes 2,3,4; set to 0000 in classes 0 and 1.
zzzz (bits 4 to 1): used to signal the CDT in classes 2,3,4; set to 1111 in class 1.
NF: not available when the non explicit flow control option is selected.
NRC: not available when the receipt confirmation option is selected.

Note – These codes are already in use in related protocols defined by standards organisations other than CCITT/ISO.

Note 5 – When the encoding of a TPDU is represented using a diagram in this clause, the following representation is used:

a) octets are shown with the lowest numbered octet to the left; higher numbered octets being further to the right;

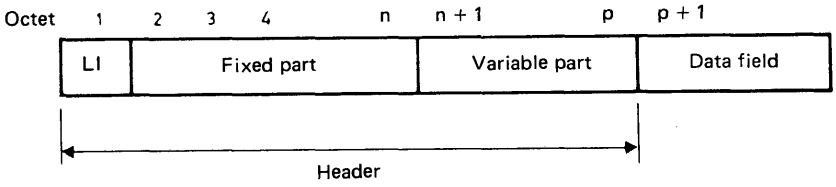
b) within an octet, bits are shown with bit 8 to the left and bit 1 to the right.

TPDUs shall contain, in the following order:

a) the header, comprising:
   1) the length indicator (LI) field,
   2) the fixed part,
   3) the variable part, if present;

b) the data field, if present.

This strucutre is illustrated as follows:



13.2.1 *Length indicator field*

This field is contained in the first octet of the TPDUs. The length is indicated by a binary number, with a maximum value of 254, (1111 1110). The length indicated shall be the header length in octets including parameters, but excluding the length indicator field and user data, if any. The value 255 (1111 1111) is reserved for possible extensions.

If the length indicated exceeds, or is equal to, the size of the NS user data which is present, this is a protocol error.

13.2.2 *Fixed part*

13.2.2.1 *General*

The fixed part contains frequently occurring parameters including the code of the TPDU. The length and the structure of the fixed part are defined by the TPDU code and in certain cases by the protocol class and the formats in use (normal or extended).

If any of the parameters of the fixed part have an invalid value, or if the fixed part cannot be contained within the header (as defined by LI), this is a protocol error.

Note – In general the TPDU code defines unambiguously the fixed part. However, different variants may exist for the same TPDU code (see normal and extended formats).

13.2.2.2 *TPDU code*

This field contains the TPDU code and is contained in octet 2 of the header. It is used to define the structure of the remaining header. This field is a full octet except in the following cases:

1110 xxxx	Connection request
1101 xxxx	Connection confirm
0101 xxxx	Reject
0110 xxxx	Data acknowledgement

where xxxx (bits 4-1) is used to signal the CDT.

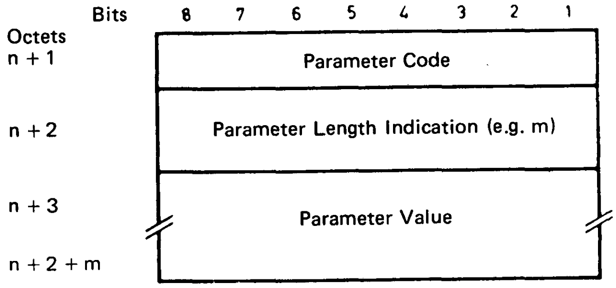
Only those codes defined in § 13.1 are valid.

13.2.3 *Variable part*

The variable part is used to define less frequently used parameters. If the variable part is present, it shall contain one or more parameters.

Note – The number of parameters that may be contained in the variable part is indicated by the length of the variable part which is LI minus the length of the fixed part.

Each parameter contained within the variable part is structured as follows:



- The parameter code field is coded in binary.

  Note – Without extensions, it provides a maximum number of 255 different parameters. However, as noted below, bits 8 and 7 cannot take every possible value, so the practical maximum number of different parameters is less. Parameter code 1111 1111 is reserved for possible extensions of the parameter code.

- The parameter length indication indicates the length, in octets, of the parameter value field.

  Note – The length is indicated by a binary number, *m*, with a theoretical maximum value of 255. The practical maximum value of *m* is lower. In the case of a single parameter contained within the variable part, two octets are required for the parameter code and the parameter length indication itself. Thus, the value of *m* is limited to 248. For larger fixed parts of the header and for each succeeding parameter, the maximum value of *m* decreases.

- The parameter value field contains the value of the parameter identified in the parameter code field.
- No parameter codes use bits 8 and 7 with the value 00.
- The parameters defined in the variable part may be in any order. If any parameter is duplicated, then the later value shall be used. A parameter not defined in this Recommendation shall be treated as a protocol error in any received TPDU except a CR TPDU. In a CR TPDU, it shall be ignored. If the responding transport entity selects a class for which a parameter of the CR TPDU is not defined, it may ignore this parameter, except the class and option and alternative protocol class parameters which shall always be interpreted. A parameter defined in this Recommendation, but having an invalid value, shall be treated as a protocol error in any received TPDU except a CR TPDU. In a CR TPDU, it shall be treated as a protocol error if it is the class and option parameter or the alternative class parameter or the additional option selection parameter; otherwise, it shall be either ignored or treated as a protocol error.

13.2.3.1 *Checksum parameter (Class 4 only)*

All TPDU types may contain a 16-bit checksum parameter in their variable part. This parameter shall be present in a CR TPDU and shall be present in all other TPDUs except when the non-use of checksum option is selected. It is encoded as follows:

Parameter Code:	1100 0011
Parameter Length:	2
Parameter Value:	Result of checksum algorithm. This algorithm is specified in § 6.17.

13.2.4 *Data field*

This field contains transparent user data. Restrictions on its size are noted for each TPDU.

13.3 *Connection request (CR) TPDU*

The length of the CR TPDU shall not exceed 128 octets.

13.3.1 *Structure*

The structure of the CR TPDU is as follows:

1 2 3 4 5 6 7 8 p p+1

LI	CR 1110	CDT	DST-REF 00000000 | 00000000	SRC-REF	CLASS, OPTIONS	Variable part	User data

13.3.2 *LI*

See § 13.2.1.

13.3.3 *Fixed part (octets 2 to 7)*

The structure of this part shall contain:

a) CR: Connection request code: 1110 Bits 8-5 of octet 2.

b) CDT: Initial credit allocation (set to 0000 in Classes 0 and 1 when specified as preferred class). Bits 4-1 of octet 2.

c) DST-REF: Set to zero.

d) SRC-REF: Reference selected by the transport entity initiating the CR TPDU to identify the requested transport connection.

e) CLASS, OPTIONS: Bits 8-5 of octet 7 defines the preferred transport protocol class to be operated over the requested transport connection. This field shall take one of the following values:

0000 Class 0
0001 Class 1
0010 Class 2
0011 Class 3
0100 Class 4.

The CR contains the first choice of class in the fixed part. Second and subsequent choices are listed in the variable part if required.

Bits 4-1 of octet 7 define options to be used on the requested transport connection as follows:

Bit	Option
4	0 always
3	0 always
2	= 0 use of normal formats in all classes = 1 use of extended formats in Classes 2, 3 ,4
1	= 0 use of explicit flow control in Class 2 = 1 no use of explicit flow control in Class 2

Bits related to options particular to a class are not meaningful if that class is not proposed and may take any value.

Note 1 – The connection establishment procedure (see § 6.5) does not permit a given CR TPDU to request use of transport expedited data transfer service (additional option parameter) and no use of explicit flow control in Class 2 (bit 1 = 1).

Note 2 – Bits 4 to 1 are always zero in Class 0 and have no meaning.

13.3.4 *Variable part (octets 8 to p)*

The following parameters are permitted in the variable part:

a) *Transport Service Access Point identifier (TSAP-ID)*

Parameter code: 1100 0001 for the identifier of the calling TSAP;

1100 0010 for the identifier of the called TSAP.

Parameter length: not defined in this Recommendation.

Parameter value: identifier of the calling or called TSAP respectively.

If a TSAP-ID is given in the request, it may be returned in the confirmation.

b) *TPDU size*

This parameter defines the proposed maximum TPDU size (in octets, including the header) to be used over the requested transport connection. The coding of this parameter is:

Parameter code:	1100 0000
Parameter length:	1 octet

Parameter value:

0000 1101	8192 octets (not allowed in Class 0)
0000 1100	4096 octets (not allowed in Class 0)
0000 1011	2048 octets
0000 1010	1024 octets
0000 1001	512 octets
0000 1000	256 octets
0000 0111	128 octets

Default value is 0000 0111 (128 octets).

c) *Version number* (not used if Class 0 is the preferred class)

Parameter code:	1100 0100
Parameter length:	1 octet
Parameter value:	0000 0001

Default value is 0000 0001 (not used in Class 0).

d) *Protection parameters* (not used if Class 0 is the preferred class)

This parameter is user defined.

Parameter code:	1100 0101
Parameter length:	user defined
Parameter value:	user defined.

e) *Checksum* (used only if Class 4 is the preferred class) (see § 13.2.3.1)

This parameter shall always be present in a CR TPDU requesting Class 4, even if the checksum selection parameter is used to request non-use of the checksum facility.

f) *Additional option selection* (not used if Class 0 is the preferred class)

This parameter defines the selection to be made as to whether or not additional options are to be used.

Parameter code:	1100 0110
Parameter length:	1

Parameter value:

Bit	Option
4	= 1 Use of network expedited in Class 1 = 0 Non use of network expedited in Class 1
3	= 1 Use of receipt confirmation in Class 1 = 0 Use of explicit AK variant in Class 1
2	= 0 16-bit checksum defined in § 6.17 is to be used in Class 4 = 1 16-bit checksum defined in § 6.17 is not to be used in Class 4
1	= 1 Use of transport expedited data transfer service = 0 No use of transport expedited data transfer service

Default value is 0000 0001.

Bits 8 to 5 shall be set to zero when sending the TPDU and ignored upon receipt.

Bits related to options particular to a class are not meaningful if that class is not proposed and may take any value.

g) *Alternative protocol class(es)* (not used if Class 0 is the preferred class)

Parameter code:	1100 0111
Parameter length:	n
Parameter value:	encoded as a sequence of single octets. Each octet is encoded as for octet 7 but with bits 4-1 set to zero (i.e. not alternative option selections permitted).

h) *Acknowledge time* (used only if Class 4 is the preferred class)

This parameter conveys the maximum acknowledge time A_L to the remote transport entity. It is an indication only, and is not subject to negotiation (see § 12.2.1.1.3).

Parameter code:	1000 0101
Parameter length:	2
Parameter value:	*n*, a binary number, where *n* is the maximum acknowledge time, expressed in milliseconds.

i) *Throughput* (not used if Class 0 is the preferred class)

Parameter code:	1000 1001
Parameter length:	12 or 24
Parameter value:	
1st 12 octets:	maximum throughput, as follows:
1st 3 octets:	target value, calling-called user direction;
2nd 3 octets:	min. acceptable, calling-called user direction;
3rd 3 octets:	target value, called-calling user direction;
4th 3 octets:	min. acceptable, called-calling user direction.
2nd 12 octets (optional):	average throughput, as follows:
5th 3 octets:	target value, calling-called user direction;
6th 3 octets:	min. acceptable, calling-called user direction;
7th 3 octets:	target value, called-calling user direction;
8th 3 octets:	min. acceptable, called-calling user direction.

Where the average throughput is omitted, it is considered to have the same value as the maximum throughput.

Values are expressed in octets per second.

j) *Residual error rate* (not used if Class 0 is the preferred class)

Parameter code:	1000 0110
Parameter length:	3
Parameter value:	
1st octet:	target value, power of 10
2nd octet:	min. acceptable, power of 10
3rd octet:	TSDU size of interest, expressed as a power of 2.

k) *Priority* (not used if Class 0 is the preferred class)

Parameter code:	1000 0111
Parameter length:	2
Parameter value:	integer (0 is the highest priority).

l) *Transit delay* (not used if Class 0 is the preferred class)

Parameter code: 1000 1000

Parameter length: 8

Parameter value:

1st 2 octets: target value, calling-called user direction;

2nd 2 octets: max. acceptable, calling-called user direction;

3rd 2 octets: target value, called-calling user direction;

4th 2 octets: max. acceptable, called-calling user direction.

Values are expressed in milliseconds, and are based upon a TSDU size of 128 octets.

m) *Reassignment time* (not used if Class 0, 2 or 4 is the preferred class)

This parameter conveys the Time to Try Reassignment/Resynchronization (TTR) which will be used when following the procedure for reassignment after failure (see § 6.12).

Parameter code: 1000 1011

Parameter length: 2

Parameter value: *n*, a binary number, where *n* is the TTR value expressed in seconds.

13.3.5 *User data (octets p+1 to the end)*

No user data are permitted in Class 0, and are optional in the other classes. Where permitted, it may not exceed 32 octets.

13.4 *Connection confirm (CC) TPDU*

13.4.1 *Structure*

The structure of the CC TPDU is as follows:

1 2 3 4 5 6 7 8 p p+1

LI	CC 1101	CDT	DST-REF	SRC-REF	CLASS, OPTIONS	Variable part	User data

13.4.2 *LI*

See § 13.2.1.

13.4.3 *Fixed part (octets 2 to 7)*

The fixed part shall contain:

a) CC: Connection confirm code: 1101. Bits 8-5 of octet 2.

b) CDT: Initial credit allocation (set to 0000 in Classes 0 and 1). Bits 4-1 of octet 2.

c) DST-REF: Reference identifying the requested transport connection at the remote transport entity.

d) SRC-REF: Reference selected by the transport entity initiating the CC TPDU to identify the confirmed transport connection.

e) CLASS, OPTIONS: Defines the selected transport protocol class and options to be operated over the accepted transport connection according to the negotiation rules specified in § 6.5.

13.4.4 *Variable part (octet 8 to p)*

The parameters are defined in § 13.3.4 and are subject to the constraints stated in § 6.5 (connection establishment). Parameters ruled out by selection of an alternative class and option shall not be present.

13.4.5 *User data (octets p+1 to the end)*

No user data are permitted in Class 0, and are optional in the other classes. Where permitted, it may not exceed 32 octets. The user data are subject to the constraints of the negotiation rules of § 6.5.

13.5 *Disconnect request (DR) TPDU*

13.5.1 *Structure*

The strucutre of the DR TPDU is as follows:

1	2	3 4	5 6	7	8 p	p+1
LI	DR 1000 0000	DST-REF	SRC-REF	REASON	Variable part	User data

13.5.2 *LI*

See § 13.2.1.

13.5.3 *Fixed part (octets 2 to 7)*

The fixed part shall contain:

a) DR: Disconnect request code: 1000 0000.

b) DST-REF: Reference identifying the transport connection at the remote transport entity.

c) SRC-REF: Reference identifying the transport connection at the transport entity initiating the TPDU. Value zero when reference is unassigned.

d) REASON: Defines the reason for disconnecting the transport connection. This field shall take one of the following values:

The following values can be used for Classes 1 to 4:

1) 128 + 0 – Normal disconnect initiated by the session entity;

2) 128 + 1 – Remote transport entity congestion at connect request time;

3) * 128 + 2 – Connection negotiation failed (i.e. proposed class(es) not supported);

4) 128 + 3 – Duplicate source reference detected for the same pair of NSAPs;

5) 128 + 4 – Mismatched references;

6) 128 + 5 – Protocol error;

7) 128 + 6 – Not used;

8) 128 + 7 – Reference overflow;

9) 128 + 8 – Connection request refused on this network connection;

10) 128 + 9 – Not used;

11) 128 + 10 – Header or parameter length invalid;

The following values can be used for all classes:

12) 0 – Reason not specified;

13) 1 – Congestion at TSAP;

14) * 2 – Session entity not attached to TSAP;

15) * 3 – Address unknown.

Note – Reasons marked with "*" may be reported to the TS-User as "persistent", other reasons as "transient".

13.5.4 *Variable part (octets 8 to p)*

The variable part may contain:

a) *Additional information* related to the clearing of the connection.

Parameter code: 1110 0000

Parameter length: Any value provided that the length of the DR TPDU does not exceed the maximum agreed TPDU size or 128 when the DR TPDU is used during the connection refusal procedure.

Parameter value: Additional information. The content of this field is user defined.

b) *Checksum*: shall be pesent if the condition in § 13.2.3.1 applies.

13.5.5 *User data (octets p+1 to the end)*

This field shall not exceed 64 octets and is used to carry TS-user data. The successful transfer of this data is not guaranteed by the transport protocol. When a DR TPDU is used in Class 0, it shall not contain this field.

13.6 *Disconnect confirm (DC) TPDU*

This TPDU shall not be used in Class 0.

13.6.1 *Structure*

The strucutre of the DC TPDU shall be as follows:

1	2	3 4	5 6	7 p
LI	DC 1100 0000	DST-REF	SRC-REF	Variable part

13.6.2 *LI*

See § 13.2.1.

13.6.3 *Fixed part (octets 2 to 6)*

The fixed part shall contain:

a) DC: Disconnect confirm code: 1100 0000.

b) DST-REF: See § 13.4.3.

c) SRC-REF: See § 13.4.3.

13.6.4 *Variable part*

The variable part shall contain the checksum parameter if the condition in § 13.2.3.1 applies.

13.7 *Data (DT) TPDU*

13.7.1 *Structure*

Depending on the class and the option, the DT TPDU shall have one of the following structures:

a) *Normal format for Classes 0 and 1*

1	2	3	4 ... end
LI	DT 1111 0000	TPDU-NR and EOT	User data

b) *Normal format for Classes 2, 3 and 4*

1	2	3 4	5	6 p	p+1 ... end
LI	DT 1111 0000	DST-REF	TPDU-NR and EOT	Variable part	User data

c) *Extended Format for use in Classes 2, 3 and 4 when selected during connection establishment*

1	2	3 4	5 6 7 8	9 p	p+1 ... end
LI	DT 1111 0000	DST-REF	TPDU-NR and EOT	Variable part	User data

13.7.2 *LI*

See 13.2.1.

13.7.3 *Fixed part*

The fixed part shall contain:

a) DT: Data transfer code: 1111 0000.

b) DST-REF: See § 13.4.3.

c) EOT: When set to ONE, indicates that the current DT TPDU is the last data unit of a complete DT TPDU sequece (end of TSDU). EOT is bit 8 of octet 3 in Classes 0 and 1 and bit 8 octet 5 for normal formats for Classes 2, 3 and 4.

d) TPDU-NR: TPDU Send Sequence Number (zero in Class 0). May take any value in Class 2 without explicit flow control. TPDU-NR is bits 7-1 of octet 3 for Classes 0 and 1, bits 7-1 of octet 5 for normal formats in Classes 2, 3 and 4 and bits 7-1 of octet 5 together with octets 6, 7 and 8 for extended formats.

Note – Depending on the class, the fixed part of the DT TPDU uses the following octets:

Classes 0 and 1:	Octets 2 to 3.
Classes 2, 3, 4 normal format:	Octets 2 to 5.
Classes 2, 3, 4 extended format:	Octets 2 to 8.

13.7.4 *Variable part*

The variable part shall contain the checksum parameter if the condition in § 13.2.3.1 applies.

13.7.5 *User data field*

This field contains data of TSDU being transmitted.

Note – The length of this field is limited to the negotiated TPDU size for this transport connection minus 3 octets in Classes 0 and 1 and minus 5 octets (normal header format) or 8 octets (extended header format) in other classes. The variable part, if present, may further reduce the size of the user data field.

13.8 *Expedited data (ED) TPDU*

The ED TPDU shall not be used in Class 0 or in Class 2 when the "no explicit flow control" option is selected, or when the expedited data transfer service has not been selected for the connection.

13.8.1 *Structure*

Depending on the format negotiated at connection establishment, the ED TPDU shall have one of the following structures:

a) Normal format (Classes, 1, 2, 3, 4)

1	2	3 4	5	6 p	p+1 ... end
LI	ED 0001 0000	DST-REF	ED-TPDU-NR and EOT	Variable part	User data

b) Extended format (for use in Classes 2, 3, 4 when selected during connection establishment)

1	2	3 4	5 6 7 8	9 p	p+1 ... end
LI	ED 0001 0000	DST-REF	ED-TPDU-NR and EOT	Variable part	User data

13.8.2 *LI*

See § 13.2.1.

13.8.3 *Fixed part*

The fixed part shall contain:

a) ED: Expedited data code: 0001 0000.

b) DST-REF: See § 13.4.3.

c) ED-TPDU-NR: Expedited TPDU identification number. ED-TPDU-NR is used in Classes 1, 3 and 4 and may take any value in Class 2. Bits 7-1 of octet 5 for normal formats and bits 7-1 of octet 5 together with octets 6, 7 and 8 for extended formats.

d) EOT: end of TSDU always set to 1 (bit 8 of octet 5).

Note – Depending on the format, the fixed part shall be either octets 2 to 5 or 2 to 8.

13.8.4 *Variable part*

The variable part shall contain the checksum parameter if the condition in § 13.2.3.1 applies.

13.8.5 *User data field*

This field contains an expedited TSDU (1 to 16 octets).

13.9 *Data acknowledgement (AK) TPDU*

This TPDU shall not be used for Class 0; for Class 2 when the "no explicit flow control" option is selected; for Class 1 when the network receipt confirmation option is selected.

13.9.1 *Structure*

Depending on the class and option agreed, the AK TPDU shall have one of the following structures:

a) *Normal format (Classes 1, 2, 3, 4)*

1 2 3 4 5 6 p

LI	AK 0110	CDT	DST-REF	YR-TU-NR	Variable part

b) *Extended format (for use in Classes 2, 3, 4 when selected during connection establishment)*

1 2 3 4 5 6 7 8 9 10 11 p

LI	AK 0110 0000	DST-REF	YR-TU-NR	CDT	Variable part

13.9.2 *LI*

See § 13.2.1.

13.9.3 *Fixed part*

The fixed part shall contain (in octets 2 to 5 when normal format is used, 2 to 10 otherwise) the following parameters:

a) AK: Aknowledgement code: 0110.

b) CDT: Credit value (set to 1111 in Class 1) bits 4-1 of octet 2 for normal formats and octets 9 and 10 for extended formats.

c) DST-REF: See § 13.4.3.

d) YR-TU-NR: Sequence number indicating the next expected DT TPDU number. For normal formats, bits 7-1 of octet 5; bit 8 of octet 5 is not significant and shall take the value 0. For extended formats, bits 7-1 of octet 5 together with octets 6, 7 and 8; bit 8 octet 5 is not significant and shall take the value 0.

13.9.4 *Variable part*

The variable part contains the following parameters:

a) *Checksum*: shall be present if the condition in § 13.2.3.1 applies.

b) *Sub-sequence number* (when optionally used under the conditions defined in Class 4).

This parameter is used to ensure that AK TPDUs are processed in correct sequence. If it is absent, this is equivalent to transmitting the parameter with a value of zero.

Parameter code:	1000 1010.
Parameter length:	2.
Parameter value:	16-bit sub-sequence number.

c) *Flow control confirmation (when optionally used under the conditions defined in Class 4)*

This parameter contains a copy of the information received in an AK TPDU, to allow the transmitter of the AK TPDU to be certain of the state of the receiving transport entity (see § 12.2.3.9).

Parameter code:	1000 1100.
Parameter length:	8.
Parameter value:	defined as follows:

1) Lower window edge (32 bits)

Bit 8 of octet 1 of the parameter value field is set to zero, the remainder contain the YR-TU-NR value of the received AK TPDU. When normal format has been selected, only the least significant seven bits (bits 1 to 7 of octet 4 of the parameter value field) of this field are significant.

2) Your sub-sequence (16 bits).

Contains the value of the sub-sequence parameter of the received AK TPDU, or zero if this parameter was not present.

3) Your credit (16 bits).

Contains the value of the CDT field of the received AK TPDU. When normal format has been selected, only the least significant four bits (bits 1 to 4 of octet 2 of the parameter value field) of this field are significant.

13.10 *Expedited data acknowledgment (EA)*

The EA TPDU shall not be used for Class 0, or for Class 2 when the "no explicit flow control" option is selected, or when the expedited data transfer service has not been selected for the connection.

13.10.1 *Structure*

Depending on the option (normal or extended format) the TPDU structure shall be:

a) Normal format (Classes 1, 2, 3, 4)

1 2 3 4 5 6 p

LI	EA 0010 0000	DST-REF	YR-TU-NR	Variable part

b) Extended format (for use in Classes 2, 3, 4 if selected during connection establishment)

1 2 3 4 5 6 7 8 9 p

LI	EA 0010 0000	DST-REF	YR-TU-NR	Variable part

13.10.2 *LI*

See § 13.2.1.

13.10.3 *Fixed part*

The fixed part shall contain (in octets 2 to 5 when normal format is used, in octets 2 to 8 otherwise):

a) EA: Expedited acknowledgement code: 0010 0000.

b) DST-REF: See § 13.4.3.

c) YR-EDTU-NR: Identification of the ED TPDU being acknowledged. May take any value in Class 2. For normal formats bits 7-1 of octet 5; bit 8 of octet 5 is not significant and shall take the value 0. For extended formats, bits 7-1 of octet 5 together with octets 6, 7 and 8; bit 8 of octet 5 is not significant and shall take the value 0.

13.10.4 *Variable part*

The variable part shall contain the checksum parameter if the condition in § 13.2.3.1 applies.

13.11 *Reject (RJ) TPDU*

The RJ TPDU shall not be used in Classes 0, 2 and 4.

13.11.1 *Structure*

The RJ TPDU shall have one of the following formats:

a) Normal format (Classes 1 and 3)

1	2		3 4	5
LI	RJ 0101	CDT	DST-REF	YR-TU-NR

b) Extended format (for use in Class 3 if selected during connection establishment)

1	2	3 4	5 6 7 8 9	10
LI	RJ 0101 0000	DST-REF	YR-TU-NR	CDT

13.11.2 *LI*

See § 13.2.1.

13.11.3 *Fixed part*

The fixed part shall contain (in octets 2 to 5 when normal format is used, in octets 2 to 10 otherwise):

a) RJ: Reject code: 0101. Bits 8-5 of octet 2.

b) CDT: Credit value (set to 1111 in Class 1). Bits 4-1 of octet 2 for normal formats and octets 9 and 10 for extended formats.

c) DST-REF: See § 13.4.3.

d) YR-TU-NR: Sequence number indicating the next expected TPDU from which retransmussion should occur. For normal formats, bits 7-1 of octet 5; bit 8 of octet 5 is not significant and shall take the value 0. For extended formats, bits 7-1 of octet 5 together with octets 6, 7 and 8; bit 8 of octet 5 is significant and shall take the value 0.

13.11.4 *Variable part*

There is no variable part for this TPDU type.

13.12 *TPDU error (ER) TPDU*

13.12.1 *Structure*

1	2	3	4	5	6	p
LI	ER 0111 0000	DST-REF		Reject cause	Variable part	

13.12.2 *LI*

See § 13.2.1.

13.12.3 *Fixed part*

The fixed part shall contain:

a) ER: TPDU error code: 0111 0000.

b) DST-REF: See § 13.4.3.

c) Reject cause:

0000 0000	Reason not specified.
0000 0001	Invalid parameter code.
0000 0010	Invalid TPDU type.
0000 0011	Invalid parameter value.

13.12.4 *Variable part*

The variable part may contain the following parameters:

a) *Invalid TPDU*

Parameter code:	1100 0001
Parameter length:	Number of octets of the value field;
Parameter value:	Contains the bit pattern of the rejected TPDU header up to and including the octet which caused the rejection. This parameter is mandatory in Class 0.

b) *Checksum*: shall be present if the condition in § 13.2.3.1 applies.

14 Conformance

14.1 A system claiming to implement the procedures specified in this Recommendation shall comply with the requirements in §§ 14.2-14.4.

14.2 The system shall implement Class 0.

14.3 If the system implements Class 3 or Class 4, it shall also implement Class 2.

14.4 For each class which the system claims to implement, the system shall be capable of:

a) initiating CR TPDUs or responding to CR TPDUs with CC TPDUs or both. Where a CR TPDU proposing Class 2, 3 or 4 is initiated, Class 0 shall be explicitly indicated as an alternative class except if there is already one (or several) transport connection(s) assigned to the network connection (multiplexing being possible);

b) responding to any other TPDU and operating network service in accordance with the procedures for the class;

c) operating all the procedures for the class listed as mandatory in Table 9/X.224;

d) operating those procedures for the class listed as optional in Table 9/X.224 for which conformance is claimed;

e) handing all TPDUs of lengths up to the lesser value of:

   1) the maximum length for the class [see § 13.3.4 b)];

   2) the maximum for which conformance is claimed (see Note 2).

Note 1 – The procedures or classes 0-4 are specified in §§ 8-12 respectively. The procedures refer to the elements of procedures specified in § 6.

Note 2 – The requirement in § 14.4.e indicates that TPDU sizes of 128 octets are always implemented.

14.5 Claims of conformance shall state:

a) which class or classes of protocol are implemented;

b) whether the system is capable of initiating or responding to CR TPDUs or both;

c) which of the procedures listed as optional in Table 9/X.224 are implemented;

d) for each class, the maximum size of TPDU implemented; the value shall be chosen from the following list and all values in the list which are less than this maximum shall be implemented [see § 13.3.4 b)]: 128, 256, 512, 1024, 2048, 4096 or 8192 octets.

TABLE 9/X.224

Provision of options

Procedure	Class 0	Class 1	Class 2	Class 3	Class 4
TPDU with checksum TPDU without checksum	NA M	NA M	NA M	NA M	M O
Expedited data transfer No expedited data transfer	NA M	M M	M M	M M	M M
Flow control in Class 2 No flow control in Class 2	NA NA	NA NA	M O	NA NA	NA NA
Normal formats Extended formats	M NA	M NA	M O	M O	M O
Use of receipt confirmation in Class 1 No use of receipt confirmation in Class 1	NA NA	O M	NA NA	NA NA	NA NA
Use of network expedited in Class 1 No use of network expedited in Class 1	NA NA	O M	NA NA	NA NA	NA NA

NA: not applicable

M: mandatory

O: optional

ANNEX A

(to Recommendation X.224)

State tables

This Annex provides a more precise description of the protocol.

The state tables also define the mapping between service and protocol events that TS-users can expect.

This Annex describes the transport protocol in terms of state tables. The state tables show the state of a transport connection, the events that occur in the protocol, the actions taken, and the resultant state.

The state tables only describe the operation of a single transport connection. They do not necessarily describe all possible combinations of sequences of events at the transport and network service boundaries, nor do they describe the exact mapping between TPDUs and NSDUs.

A.1 *Conventions*

A.1.1 The incoming events are represented in the state tables by their abbreviated name as defined in Table A-1/X.224;

A.1.2 The states are represented in the tables by their abbreviated name as defined in Table A-2/X.224;

A.1.3 The intersection of each state and event which is invalid is left blank. The action to be taken in this case is one of the following:

a) for an event related to the transport service (i.e. coming from the TS-user), take no action;

b) for an event related to a received TPDU, follow the procedure for treatment of protocol errors (see § 6.22) if the state of the supporting network connection makes it possible;

c) for an event falling into neither of the above categories (including those which are impossible by the definition of the behaviour of the transport entity or NS-provider) take no action.

A.1.4 At each intersection of state and event which is valid the state tables specify an action which may include one of the following:

a) one action constituted of a list of any number of outgoing events (none, one, or more) given by their abbreviated name defined in Table A-3/X.224 followed by the abbreviated name of the resultant state (see Table A-2/X.224);

b) conditional actions separated by a semi-colon (;). Each conditional action contains a predicate followed by a colon (:) and by an action as defined in § A.1.4 a). The predicates are boolean expressions given by their abbreviated name and defined in the clauses related to the state tables of each class. Only the action corresponding to the predicate which is true is to be taken.

A.1.5 The state tables also include:

a) informal comments giving explanatory material;

b) references to notes using the following notation: (note number);

c) references to other actions defined in separate tables using the following notation: (action number).

A.2 *General*

Table A-1/X.224 specifies the names and abbreviated names of the incoming events, classified as TS-user events, NS-provider events or TPDU events.

Table A-2/X.224 specifies the names and abbreviated names of the states.

Table A-3/X.224 specifies the names and abbreviated names of the outgoing events classified as TS-provider events, NS-user events or TPDU events.

TABLE A-1/X.224

Incoming events

Abbreviated name	Category	Name
TCONreq	TS-user	T-CONNECT request primitive
TCONresp	TS-user	T-CONNECT response primitive
TDTreq	TS-user	T-DATA request primitive
TEXreq	TS-user	T-EXPEDITED DATA request primitive
TDISreq	TS-user	T-DISCONNECT request primitive
NDISind	NS-provider	N-DISCONNECT indication primitive
NCONconf	NS-provider	N-CONNECT confirm primitive
NRSTind	NS-provider	N-RESET indication primitive
CR	TPDU	Connection request TPDU
CC	TPDU	Connection confirm TPDU
DR	TPDU	Disconnect request TPDU
DC	TPDU	Disconnect confirm TPDU
AK	TPDU	Data acknowledgement TPDU
EA	TPDU	Expedited data acknowledgement TPDU
DT	TPDU	Data TPDU
ED	TPDU	Expedited data TPDU
ER	TPDU	TPDU error TPDU
RJ	TPDU	Rejet TPDU

TABLE A-2/X.224

States

Abbreviated name	Name
WFNC	Wait for network connection
WFCC	Wait for the CC TPDU
WBCL	Wait before releasing (wait for CC TPDU before sending the DR TPDU)
OPEN	Transport connection is open
CLOSING	Release in progress
WFTRESP	Wait for T-CONNECT response
CLOSED	Transport connection is closed
WFNC-R	Wait for network connection and reassignment in progress
WFCC-R	Wait for CC TPDU and reassignment in progress
WBCL-R	Wait before releasing and reassignment in progress
OPEN-R	Open and reassignment in progress
OPEN-WR	Open and wait for reassignment
CLOSING-R	Release in progress and reassignment in progress
CLOSING-WR	Release in progress and wait for reassignment
WFTRESP-WR	Wait for T-CONNECT response and wait for reassignment
WBCL-WR	Wait before releasing and wait for reassignment
WBOC	Wait before open complete (CC is unacknowledged)
WBOC-WR	Wait before open complete and wait for reassignment
CLOSING BOC	Wait before open complete and release in progress
CLOSING BOC-WR	Idem and wait for reassignment
AKWAIT	Wait for acknowledgement of CC TPDU
REFWAIT	Waiting for frozen reference time

TABLE A-3/X.224

Outgoing events

Abbreviated name	Category	Name
TCONind	TS-provider	T-CONNECT indication primitive
TCONconf	TS-provider	T-CONNECT confirm primitive
TDTind	TS-provider	T-DATA indication primitive
TEXind	TS-provider	T-EXPEDITED DATA indication primitive
TDISind	TS-provider	T-DISCONNECT indication primitive
NDISreq	NS-user	N-DISCONNECT request primitive
NRSTresp	NS-user	N-RESET response primitive
NCONreq	NS-user	N-CONNECT request primitive
CR	TPDU	Connection request TPDU
CC	TPDU	Connection confirm TPDU
DR	TPDU	Disconnect request TPDU
DC	TPDU	Disconnect confirm TPDU
AK	TPDU	Data acknowledgement TPDU
EA	TPDU	Expedited data acknowledgement TPDU
DT	TPDU	Data TPDU
ED	TPDU	Expedited data TPDU
ER	TPDU	TPDU error TPDU
RJ	TPDU	Reject TPDU

A.3 *State tables for Classes 0 and 2*

This section provides a more precise description of a transport entity for a transport connection of Class 0 or Class 2.

The description uses predicates defined in Table A-4/X.224, and specific actions defined in Table A-5/X.224.

The description does not include a complete specification of the data transfer procedures but makes reference to the specification of the classes (see §§ 8 and 10). Table A-6/X.224 gives the state automata for Classes 0 and 2.

TABLE A-4/X.224

Predicates for classes 0 and 2

Name	Description
P0	T-CONNECT request unacceptable
P1	Unacceptable CR TPDU
P2	No network connection available
P3	Network connection available and open
P4	Network connection available and open in progress
P5	Class is class 0 (class selected in CC)
P6	Unacceptable CC
P7	Class is class 2
P8	Acceptable CC
P9	Class 4 CR

TABLE A-5/X.224

Specific actions for classes 0 and 2

Name	Description
[1]	If the network connection is not used by another transport connection assigned to it, it may be released.
[2]	See § 6.22 (receipt of an ER TPDU)
[3]	See data transfer procedures of the class
[4]	See expedited data transfer procedures of the class
[5]	An N-RESET response has to be issued once for the network connection if the network connection has not been released. In class·0, an N-DISCONNECT request has to be issued.

TABLE A-6/X.224

State table for classes 0 and 2 (part 1 of 2)

STATE EVENT	WFNC	WFCC	WBCL (Class 2 only)	OPEN	CLOSING (Class 2 only)	WFTRESP	CLOSED
TCONreq							P0: TDISind CLOSED; P2: NCONreq WFNC; P3: CR WFCC; P4: WFNC
TCONresp						CC OPEN	
TDTreq				[3] OPEN			
TEXreq	DOES NOT EXIST IN CLASS 0						
				[4] OPEN			
TDISreq	[1] CLOSED	not P7: NDISreq CLOSED P7: WBCL		P5: NDISreq CLOSED; P7: DR CLOSING		DR CLOSED	
NCONconf	CR WFCC						
NRSTind		TDISind [1] [5] CLOSED	[1] [5] CLOSED	TDISind [1] [5] CLOSED	[1] [5] CLOSED	TDISind [1] [5] CLOSED	
NDISind	TDISind CLOSED	TDISind CLOSED	CLOSED	TDISind CLOSED	CLOSED	TDISind CLOSED	
CR				P9: OPEN	P9: CLOSING	P9: WFTRESP	P1: DR (1) CLOSED; not P1: TCONind WFTRESP
DR		TDISind [1] CLOSED	[1] CLOSED	P5: (2); P7: DC TDISind CLOSED	[1] CLOSED		(4) CLOSED; DC CLOSED

TABLE A-6/X.224

State table for classes 0 and 2 (Part 2 of 2)

<table>
<tr><th>STATE
EVENT</th><th>WFNC</th><th>WFCC</th><th>WBCL (Class 2 only)</th><th>OPEN</th><th>CLOSING (Class 2 only)</th><th>WFTRESP</th><th>CLOSED</th></tr>
<tr><td rowspan="2">DC</td><td colspan="7">DOES NOT EXIST IN CLASS 0 (2)</td></tr>
<tr><td></td><td></td><td></td><td></td><td>P7: [1] CLOSED</td><td></td><td>CLOSED</td></tr>
<tr><td>CC</td><td></td><td>P8: TCONconf OPEN;
P6 and P5: TDISind NDISreq CLOSED;
P6 and P7: TDISind DR CLOSING</td><td>P5: (3) NDISreq CLOSED;
P7: DR CLOSING</td><td></td><td></td><td></td><td>DR CLOSED</td></tr>
<tr><td rowspan="2">AK</td><td colspan="7">DOES NOT EXIST IN CLASS 0 (2)</td></tr>
<tr><td></td><td></td><td></td><td>[3] OPEN</td><td>CLOSING</td><td></td><td>CLOSED</td></tr>
<tr><td rowspan="2">EA</td><td colspan="7">DOES NOT EXIST IN CLASS 0 (2)</td></tr>
<tr><td></td><td></td><td></td><td>[4] OPEN</td><td>CLOSING</td><td></td><td>CLOSED</td></tr>
<tr><td rowspan="2">ED</td><td colspan="7">DOES NOT EXIST IN CLASS 0 (2)</td></tr>
<tr><td></td><td></td><td></td><td>[4] OPEN</td><td>CLOSING</td><td></td><td>CLOSED</td></tr>
<tr><td>DT</td><td></td><td></td><td></td><td>[3] OPEN</td><td>CLOSING</td><td></td><td>CLOSED</td></tr>
<tr><td>ER</td><td></td><td>TDISind [1] CLOSED</td><td>[1] CLOSED</td><td>[2]</td><td>[2]</td><td></td><td>CLOSED</td></tr>
</table>

Note 1 – An ER TPDU shall be sent in certain cases (see § 6.6).

Note 2 – If received, it shall be processed as a protocol error (see § 6.22).

Note 3 – A CR with Class 2 has been sent and a CC Class 0 is received.

Note 4 – If DC is not available (i.e. Class 0 only implemented), or SRC-REF is zero.

A.4 *State tables for Classes 1 and 3*

This section provides a more precise description of a transport entity for a transport connection of Class 1 or Class 3.

The description uses the predicates defined in Table A-7/X.224.

Specific actions are defined in Table A-8/X.224 and specific additional notes are given in Table A-9/X.224.

The description does not include a complete specification of the data transfer procedures but makes reference to the specification of the classes (see §§ 9 and 11). Table A-10/X.224 gives the state automata for Classes 1 and 3.

TABLE A-7/X.224

Predicates for classes 1 and 3

Name	Description
P0	T-CONNECT request unacceptable
P1	No available network connection can be used for assignment or reassignment
P2	A network connection can be used for asignment or reassignment; the network connection opening is in progress
P3	A network connection can be used for assignment or reassignment; the network connection is open
P4	TTR timer has previously run out
P5	Local choice
P6	Initiator of the transport connection
P7	Unacceptable CR TPDU
P8	TWR is running
P9	Class 4 CR
P10	Class selected in CC is class 0 or 2

TABLE A-8/X.224

Specific actions for classes 1 and 3

Name	Description
[1]	The network connection can be disconnected if not used by any transport connection assigned to it.
[2]	Process TDT request or TEX request which have been stored when waiting for reassignment (if any). If an RJ TDPU has been received, enable also data TPDU transmission (if any). If an ED TPDU was received, handle according to procedures for class if not a duplicate.
[3]	Network connection can be disconnected if not used by any transport connection and was locally opened.
[4]	Start TWR timer if not already running. Disable sending DT TPDUs until an RJ TPDU is received (see Note 3).
[5]	Stop TWR timer.
[6]	Issue an N-RESET response if not already done.
[7]	See data transfer procedure for the class.
[8]	Start TTR timer if not already running.
[9]	Stop TTR timer if running or remove information that TTR timer has run out (see Notes 1 and 2).
[10]	Store information that TTR timer has run out (see Note 1).
[11]	Store request.
[12]	See state table appropriate to class selected in CC.

Note 1 – The information is used by predicate P4.

Note 2 – This action is not performed if the transport entity is the responder of if neither reassignment nor resynchronization is in progress.

Note 3 – The method of disabling transmission of DT-TPDUs is a local matter. In Class 3 for example, it may be effected by setting credit to zero. In Class 1, this may be effected by the setting of a boolean indicator.

TABLE A-9/X.224

Specific notes for classes 1 and 3

Name	Description
(1)	Any TPDU except DR and CC having an unknown destination reference.
(2)	CC TPDU having an unknown destination reference or a mismatched source reference.
(3)	CR TPDU which is not duplicated but rejected. If CR TPDU is duplicated, ignore it.
(4)	Or send any DT or ED TPDU waiting for transmission or use N-DATA ACKNOWLEDGE request if available and selected (class 1 only).
(5)	Same as for (9) and issue a T-DISCONNECT indication.
(6)	If the resultant state is CLOSED, the reference shall be frozen except in the cases described in § 6.18.
(7)	An ER TPDU shall be sent in certain cases (see § 6.6).
(8)	Receipt of a DC TPDU is a protocol error since DC cannot be used for reassignment. It is suggested to stop the TWR timer [5] and to consider the transport connection as released (CLOSED state).
(9)	Receipt of one of these TPDUs in this state is a protocol error. It is suggested to stop the TWR timer [5], send a DR TPDU and enter the CLOSING state.
(10)	Or a DR with mismatched source reference has been received.

TABLE A-10/X.224

State table for classes 1 and 3 (part 1 of 3)
(connection – responder side)

STATE EVENT	CLOSED	WFTRESP	WFTRESP-WR	WBCL-WR	WBOC	WBOC-WR	CLOSING BOC	CLOSING BOC-WR
TDISreq		DR CLOSED (6)	WBCL-WR		DR CLOSING BOC	CLOSING BOC-WR		
TCONreq		P10: [12]; not P10: CC WBOC	WBOC-WR					
NRSTind		[4] [6] WFTRESP-WR	[6] WFTRESP-WR	[6] WBCL-WR	[4] [6] WBOC-WR	[6] WBOC-WR	[4] [6] CLOSING BOC-WR	[6] CLOSING BOC-WR
NDISind		[4] WFTRESP-WR	WFTRESP-WR	WBCL-WR	[4] WBOC-WR	WBOC-WR	[4] CLOSING BOC-WR	CLOSING BOC-WR
CR	P7: DR (3.7) CLOSED (6); not P7: TCONind WFTRESP	P9: WFTRESP	[5] WFTRESP	[5] DR CLOSED (6)	P9: WBOC	[5] CC WBOC	P9: BOC	[5] DR CLOSED (6)
DR	DC CLOSED				TDISind DC CLOSED (6)	DC [5] TDISind CLOSED (6)	CLOSED (6)	[5] DC CLOSED (6)
RJ or ED	CLOSED				OPEN [7]	[5] [2] RJ OPEN	CLOSING	[5] DR CLOSING
DC	CLOSED						CLOSED (6)	(8)
ER					TDISind DR CLOSING BOC		CLOSED (6)	
First TPDU other than CR, ER, DR, DC, ED or RJ	CLOSED				OPEN [7]		CLOSING	(9)
TWR Time-out			TDISind CLOSED (6)	CLOSED (6)		TDISind CLOSED (6)		CLOSED (6)
TDTreq					[7] WBOC	[11] WBOC-WR		
TEXreq					[7] WBOC	[11] WBOC-WR		

TABLE A-10/X.224

State table for classes 1 and 3 (part 2 of 3)
(connection – initiator side)

STATE EVENT	CLOSED	WFNC	WFNC-R	WFCC	WFCC-R	WBCL	WBCL-R
TCONreq	P0: TDISind CLOSED; not P0 and P1: NCONreq WFNC; not P0 and P2: WFNC; not P0 and P3: CR WFCC						
NCONconf		CR WFCC	CR WFCC		CR WFCC		CR WBCL
NRSTind				P4: TDISind [1] [6] CLOSED (6); not P4: CR [6] [8] WFCC		P4: [6] CLOSED [1]; not P4: CR [6] [8] WBCL	
NDISind		P1: NCONreq [2] WFNC-R [8]; P2: [5]; WFNC-R; P3: CR [8] WFCC	P1: NCONreq WFNC-R; P2: WFNC-R; P3: CR WFCC	P4: TDISind CLOSED (6); (not P4) and P1: [8] NCONreq WFCC-R; (not P4) and P2: [8] WFCC-R; (not P4) and P3: [8] CR WFCC	[2] WFCC-R	P4 or P5: [1] [9] CLOSED (6); (not P4 or P5) and P1: [8] NCONreq WBCL-R; (not P4 or P5) and P2: [8] WBCL-R; (not P4 or P5) and P3: [8] CR WBCL	P5: [9] CLOSED (6); (not P5) and P1: NCONreq WBCL-R; (not P5) and P2: WBCL-R (not P5) and P3: CR WBCL
TDISreq		[1] CLOSED (6)	[1] CLOSED (6) [9]	WBCL	P5: [1] [9] CLOSED (6); not P5: WBCL-R		
DR	(10) DC CLOSED			TDISind [1] [9] CLOSED (6)		[1] [9] CLOSED (6)	
CC	DR CLOSED			P10: [12]; not P10: TCONconf AK (4) [9] OPEN		P10: [12]; not P10: DR [9] CLOSING	
ER				TDISind [1] [9] CLOSED (6)		[1] [9] CLOSED (6)	
(1)	CLOSED						
(2)	DR CLOSED						
Time-out TTR			TDISind [1] [9] CLOSED (6)	[10]	TDISind [1] [9] CLOSED (6)	[10]	[1] CLOSED (6)

TABLE A-10/X.224

State table for classes 1 and 3 (part 3 of 3)
(open and closing states)

STATE EVENT	OPEN	OPEN-R	OPEN-WR	CLOSING	CLOSING-R	CLOSING-WR
NCONconf		RJ [2] OPEN			DR CLOSING	
TDISreq	P8: CLOSING; not P8: DR CLOSING	CLOSING-R	CLOSING-WR			
NRSTind	P6 and P4: [6] TDISind [3] CLOSED (6); P6 and not P4: [6] [2] [8] RJ OPEN; not P6: [4, 6] OPEN			P6 and P4: [6] [3] CLOSED (6) P6 and not P4: [6] [8] DR CLOSING; not P6: [4, 6] CLOSING		
NDISind	P6 and P4: TDISind CLOSED (6); (P6 and not P4) and P1: [8] NCONreq OPEN-R; (P6 and not P4) and P2: [8] [2] [8] OPEN-R; (P6 and not P4) and P3: RJ OPEN not P6: [4] OPEN-WR	P1: NCONreq OPEN-R; P2: OPEN-R; P3: [2] RJ OPEN		P6 and (P4 or P5) CLOSED (6); P6 and not (P4 or P5) and P1: [8] NCONreq CLOSING-R; P6 and not (P4 or P5) and P2: [8] CLOSING-R; P6 and not (P4 or P5) and P3: [8] DR CLOSING; not P6: [4] CLOSING-WR	P5: CLOSED (6); (not P5) and P1: NCONreq CLOSING-R; (not P5) and P2: CLOSING-R; (not P5) and P3: DR CLOSING	
RJ	P8: [5] [2] RJ OPEN; not P8: [7] [9] OPEN		RJ [5] [2] OPEN	P8: [5] DR CLOSING; not P8: [9] CLOSING		DR [5] CLOSING
Time-out TWR	TDISind CLOSED (6)		TDISind CLOSED (6)	CLOSED (6)		CLOSED (6)
DR	P8: TDISind DC (6) [5] CLOSED; not P8: TDISind DC (6) [9] CLOSED		[5] DC TDISind CLOSED (6)	P8: [5] DC (6) CLOSED; not P8: [3] [9] (6) CLOSED		[5] DC CLOSED (6)
DC				P8: (8); not P8: [3] [9] CLOSED (6)		(8)
ER	TDISind DR CLOSING		TDISind DR CLOSING	CLOSED (6)		CLOSED (6)
DT, AK or EA TPDU	[7] OPEN		(5)	CLOSING		(9)
Time-out TTR	[10]	TDISind [1] CLOSED (6)		[10]	[1] CLOSED (6)	
TDTreq	P8: [11] OPEN; not P8: [7] OPEN	[11] OPEN-R	[11] OPEN-WR			
TEXreq	P8: [11] OPEN; not P8: [7]	[11] OPEN-R	[11] OPEN-R			

A.5 *State tables for Class 4*

This section provides a more precise description of a Class 4 transport connection. Tables A-11, A-12, A-13 give the predicates, actions and notes for Class 4 respectively. Table A-14/X.224 is the state table for Class 4 transport connection.

Assumptions and notations:

a) the state of every network connection is known as being open or opening (i.e., a NCONreq has been issued and the NCONCconf is awaited);

b) for each transport connection the transport entity maintains the set of network connections to which the transport is assigned. A network connection in this set is either in open or opening state;

c) when a N-CONNECT confirmation, N-RESET indication or N-DISCONNECT indication is received this event is associated with the transport connection if the network connection belongs to the set;

d) when an N-DISCONNECT is received the network connection becomes unexisting and is therefore withdrawn from the set. When a NCONconf is received the state of the nc becomes "open";

   Note – This is not shown by explicit action in the state table. Conversely adding an nc to a set and setting the state of an nc to "opening" is shown by explicit action;

e) when the state goes back to CLOSED or REFWAIT state, it is assumed that all timers are stopped (if running), the count is set to zero and the set becomes empty;

f) when a PDU is recieved the network connection on which it has been received is assumed to be known;

g) the variable "curent-nc" is used to designate either the nc on which a TPDU has been received or the nc which has been chosen for a new assignment (either an existing one or a new one which is created);

h) we also assume the following variables:

   1) local-ref: the reference (local) of the TC is chosen when sending the CR or when accepting a CR,

   2) remote-ref: the reference of remote entity is initially set to zero and initialized when processing the CC except if the CC is ignored,

   3) SRC-REF, DST-REF: designates the corresponding field of the received TPDU,

   4) src-ref, dst-ref: designates the corresponding field of the sent TPDU,

   5) count: designates the number of times a TPDU has been sent (retransmission);

i) the data transfer phase is not completely described in the state table but refers to the main text;

j) a "spontaneous" event called "decision of making a new assignment" has been introduced. It may occur at any time provided:

   – Pl and P2 are true (see predicate table) and the remote-ref is not zero (i.e., a CR TPDU has been received or a CC TPDU has been received and processed);

k) when an N-RESET indication is received, an N-RESET response is issued.

TABLE A-11/X.224

Predicates for class 4

Name	Description
P0	T-CONNECT request is acceptable.
P1	An assignment can be done to a suitable Network Connection (either open or opening).
P2	It is possible to open a new Network Connection.
P3	Local choice.
P4	A CR TPDU has never been sent.
P5	The transport entity is the initiator and the set of Network Connections is now empty (i.e., a new assignment shall be done) or a new assignment is decided as a local choice.
P6	Local choice not to perform a new assignment if the set of Network Connections is empty (for closing state only).
P7	Count = maximum.
P8	Acceptable CR TPDU.
P9	Acceptable class 4 CC TPDU.
P10	Unacceptable classe 4 CC TPDU.
P11	CC TPDU not specifying class 4.

TABLE A-12/X.224

Specific actions for class 4

Name	Description
[0]	Set reference timer.
[1]	Count = count + 1.
[2]	Count = 0.
[3]	Set retransmission timer.
[4]	Stop retransmission timer if running.
[5]	Set window timer.
[6]	Stop window timer if running.
[7]	Set inactivity timer.
[8]	Stop inactivity timer.
[9]	Set initial credit for sending according to the received CR/CC TPDU.
[10]	Set initial credit for controlling reception according to the sent CR/CC TPDU.
[11]	Send the CR TPDU if there is a Network Connection in the open state in the set.
[12]	Add the current Network Connection to the set, if not already included.
[13]	The current Network Connection is now in the opening state.
[14]	Send the CC TPDU if a Network Connection in the open state is in the set.
[15]	Send the DR TPDU if a Network Connection in the open state is in the set. This DR TPDU is sent with SRC-REF = local-ref and DST-REF = remote-ref (may be ZERO).
[16]	Send the DR TPDU if a Network Connection in the open state is in the set. The DR TPDU is sent with SRC-REF = 0 and DST-REF = remote-ref.
[17]	Send a TPDU according to data transfer procedure.
[18]	See state table if the class specified in the CC TPDU (refer to data transfer).
[19]	See state table for the class (refer to release procedure); send a DR TPDU if the class is not 0, otherwise issue an N-DISCONNECT request.
[20]	Store request an exercise flow control to the user.
[21]	Send a DR TPDU with SRC-REF field set to zero.
[22]	Send a DC TPDU DC except if the SRC-REF field of the received DR TPDU is equal to zero.

TABLE A-13/X.224

Specific notes for class 4

Name	Description
(1)	Not possible as not set of Network Connection is associated with this Transport Connection.
(2)	It is also possible to remain in the same state (T1 is still running) until: – a CC TPDU is received which performs a new assigment, – a new assignment is tried (spontaneous event), – T1 runs out and the count is equal to the maximal value.
(3)	No new assignment was possible: if the set is empty, the Transport Entity waits until a new assignment is received, or can be locally performed (sontaneous event).
(4)	It is also possible to perform a new assignment (this may be done in triggering the event «new Network Connection assignment»).
(5)	Not a duplicated CR TPDU. If CR TPDU is duplicated, ignore it.
(6)	Since a new Network Connection is now assigned, it is recommended that the appropriate TPDU be sent on this Network Connection (if open) in order to make the remote entity aware of this assignment. It is also possible to allow the normal retransmission procedures to pause for the TPDU to be sent; however, the first TPDU available for sending should be sent on the new Network Connection.
(7)	As a local choice it is also possible to apply the following: [0], TDISind, REFWAIT.
(8)	Association to this Transport Connection is done regardless of the SRC-REF field. If SRC-REF is not zero, a DC TPDU is sent back.
(9)	At least an AK TPDU shall be sent if the Transport Entity is the initiator in order to ensure that the responder will complete is its three-way handshake.
(10)	If association has been made, and DST-REF is zero, then the DC TPDU countains a SRC-REF field set to zero.
(11)	If the CLOSING state has been entered coming from WFCC state, the remote-ref is zero – the SRC-REF field of the CC TPDU is ignored (i.e, if the DR TPDU is retransmitted, it will be with DST-REF field set to zero).
(12)	If the CLOSING state has been entered, coming from WFCC state, the remote-ref (which is zero) shall be set with SRC-REF in order to comply with the release procedure of the negotiated class.
(13)	The DR TPDU may be either repeated immediately or when T1 will run out.
(14)	If the set is empty, this event may be used as a criteria for triggering the event «new Network Connection assignment».
(15)	Previously stored T-DATA or T-EXPEDITED-DATA requests are ready for processing according to data transfer procedures.
(16)	See data transfer procedures.
(17)	When an N-RESET indication is received, an N-RESET response to be issued once independent of the state automata.

TABLE A-14/X.224

Class 4 connection/disconnection

STATE / EVENT	REFWAIT	CLOSED	WFCC	WBCL	OPEN	WFTRESP	AKWAIT	CLOSING

TCONreq		not P0: TDISind CLOSED;
		P0 & P1: [12, 1, 3, 10, 11] WFCC;
		P0 & not P1 & P2: [13, 12, 1, 3, 10] NCONreq WFCC;
		P0 & not P1 & not P2: TDISind CLOSED						

TCONresp						[3, 2, 1, 10, 14] AKWAIT		
TDISreq			P4: CLOSED; not P4 & P3 WBCL; not P4 & not P3 [4, 3, 2, 1, 15] CLOSING		[6, 8, 4, 3, 2, 1, 15] CLOSING	[16] CLOSED	[4, 3, 2, 1, 15] CLOSING	

NDISind	(1)	(1)	P1: [12, 4] WFCC; (not P1) & P2: [13, 12] NCONreq WFCC; not P1 & not P2: [0] (2) TDISind REFWAIT	P3: [0] REFWAIT; not P3 & P1: [12, 11] WBCL; not P3 & not P1 & P2: [13, 12] NCONreq WBCL; not P3 & not P1 & not P2): [0] REFWAIT	P5 & P1 [12, 17] (6) OPEN;
					P5 & not P1 & P2: [13, 12] NCONreq OPEN;
					P5 & not P1 & not P2: OPEN (3); not P5: OPEN
						WFTRESP (4)	P5 & P1: [12, 14] (6) AKWAIT; P5 & not P1 & P2: [13, 12] NCONreq AKWAIT: P5 & not P1 & not P2: AKWAIT (3); not P5: AKWAIT	P6: [0] REFWAIT; not P6 & P5 & P1: [12, 15] CLOSING (6); not P6 & P5 & not P1 & P2: [13, 12] NCONreq CLOSING; not P6 & P5 & not P1 & not P2: CLOSING (3); not P6 & not P5: CLOSING

TABLE A-14/X.224 *(continued)*

STATE / EVENT	REFWAIT	CLOSED	WFCC	WBCL	OPEN	WFTRESP	AKWAIT	CLOSING
NRSTind			(17)	(17)	(17)	(17)	(17)	(17)
TDTreq TEXreq					(16) OPEN		[20] AKWAIT	
NCONconf	(1)	(1)	CR WFCC (6)	CR WBCL (6)	[17] OPEN (16)	WFTRESP	CC AKWAIT (6)	[15] CLOSING (6)
New Network Connection Assignment					P1: [12, 17] OPEN (6); not P1 & P2: [13, 12] NCONreq OPEN	P1: [12] WFTRESP; not P1 & P2: [13, 12] NCONreq WFTRESP	P1: [12, 14] (6) AKWAIT; not P1 & P2: [13, 12] NCONreq AKWAIT	P1: [12, 15] (6) CLOSING; not P1 & P2: [13, 12] NCONreq CLOSING
Retrans-t			P7 & P3: [0] TDISind REFWAIT: P7 & not P3: [3, 2, 1, 15] TDISind CLOSING (14); not P7: [1, 3, 11] WFCC	P7 & P3: [0] REFWAIT P7 & not P3: [3, 2, 1, 15] CLOSING (14); not P7: [1, 3, 11] WBCL	P7: [6, 8, 3, 2, 1, 15] TDISind CLOSING (14); not P7: (16) (14) OPEN		P7: [3, 2, 1, 15] TDISind (14) CLOSING; not P7: [1, 3, 14] (14) AKWAIT	P7: [0] REFWAIT; not P7: [1, 3, 15] (14) CLOSING
I-t					[6, 4, 3, 2, 1, 15] TDISind CLOSING (7)			
Ref-t	CLOSED							
CR		not P8: [21] CLOSED (5); P8: [9, 12] TCONind WFTRESP (5)			[12, 8, 7] OPEN	[12] WFTREST	[12, 14] AKWAIT	[12] CLOSING (13)

TABLE A-14/X.224 *(end)*

STATE EVENT	REFWAIT	CLOSED	WFCC	WBCL	OPEN	WFTRESP	AKWAIT	CLOSING

CC	DR
	REFWAIT
		DR
		CLOSED
			P9: [12, 9, 2, 4, 5, 7, 17] TCONconf (9) OPEN;
			P10: [12, 4, 3, 2, 1, 15] TDISind CLOSING;
			P11: [18]
				P11: [19] non P11: [12, 2, 4, 3, 1, 15] CLOSING	[12, 17, 8, 7]
					(9) OPEN
								P11: [19] (12);
								not P11: [12]
								CLOSING (11)

ER	REFWAIT	CLOSED	[0] TDISind REFWAIT	[0] REFWAIT	[12, 6, 8, 4, 3, 2, 1, 15] TDISind CLOSING		[12, 4, 3, 2, 1, 15] TDISind CLOSING	[0] REFWAIT
DR	[22] REFWAIT	[22] CLOSED	(8) TDISind [0] REFWAIT	(8) [0] REFWAIT	DC (10) [0] TDISind REFWAIT	DC (10) TDISind CLOSED	DC (10) [0] TDISind REFWAIT	[0] REFWAIT
DC	REFWAIT	CLOSED						[0] REFWAIT
EA	REFWAIT	CLOSED			[12, 8, 7] OPEN (16)			[12] CLOSING (13)
DT/AK/ED	REFWAIT	CLOSED			[12, 8, 7] OPEN (16)		[12, 7] OPEN (15) (16)	[12] CLOSING (13)

ANNEX B

(to Recommendation X.224)

Transport protocol identification

B.1 *Introduction*

The transport protocol identifier is used to signal, from the initiator of the network connection to the acceptor, what transport protocol is to be used on that network connection. The procedure uses a PI TPDU, whose encoding is defined in § B.3 below. There are two ways that the PI TPDU may be conveyed, as described in § B.2.

This case of transport protocol identification only applies to the use of a network connection for a single transport protocol (such as that described in the remainder of this Recommendation). The general case, whereby the network connection can be used for several different transport protocols, whether sequentially or concurrently is for further study.

Note 1 – No negotiation of PI is currently provided for. It is for further study what mechanisms and TPDUs would be needed to support such negotiation in the future.

Note 2 – It is for further study how to achieve compatibility of this transport protocol identification with existing terminals.

B.2 *Conveyance of PI TPDU*

The PI TPDU may be conveyed either as NS-user data of the N-CONNECT request (and indication) or as N-DATA, as described in the following two subsections.

B.2.1 *N-CONNECT NS-user data available*

When a transport entity opens a new network connection for the purpose of assignment (or reassignment) the transport entity shall use the NS-user data parameter of the N-CONNECT request primitive, as follows: either

a) a PI TPDU shall be placed as the first, or only, TPDU in the NS-user data field. This TPDU explicitly indicates which protocol is to be used over the network connection; or

b) no PI TPDU is conveyed with the N-CONNECT – in this case the default is considered to apply, namely that the transport protocol specified in the remainder of this Recommendation is to be used.

Note 1 – Teletex and Group 4 facsimile terminals operating over the PSPDN will send (and thus may expect to receive) a protocol identifier of a single octet with the value "02", as specified in Reference 4. This must be taken into account when interworking with Teletex or Group 4 facsimile terminals is required.

Note 2 – It is for further study whether N-DISCONNECT NS-user data, if provided, may be used, for example in negotiation of protocol identification.

B.2.2 *No N-CONNECT NS-user data available*

For an interim period, NS-user data may not be available in the N-CONNECT request under some circumstances. Under these circumstances, the PI TPDU shall be conveyed using the N-DATA request, and shall be concatenated with the CR TPDU which requests opening of the transport connection, or the first TPDU sent on that network connection performing assignment or reassignment.

When no PI TPDU is conveyed, the default transport protocol, i.e. the protocol defined in the remainder of this Recommendation, will apply.

B.3 *Encoding of the PI TPDU*

B.3.1 *Structure*

The PI TPDU shall have the following format:

1	2	3	4	5 p
LI	PI 0000 0001	PRT-ID	SHARE	Variable part

B.3.2 *LI*

See § 13.2.1.

B.3.3 *Fixed part*

The fixed part shall contain (in octets 2 to 4):

a) PI: Protocol identification code: 0000 0001.

b) PRT-ID: Protocol Id

00000000	reserved
00000001	OSI transport protocol (remainder of this Recommendation)
00000010-01111111	reserved for other OSI protocols
10000000-11111111	reserved for private use.

c) SHARE: Sharing of network connection

00000000	no sharing
00000001	sharing allowed (for further study)
00000010-11111111	reserved.

B.3.4 *Variable part*

One optional parameter is defined – however its use is for further study:

protocol sharing

Parameter code:	1101 1111
Parameter length:	number of protocol ids
Parameter value:	list of protocol ids, one per octet

This parameter is not permitted unless SHARE has the value 00000001.

APPENDIX I

(to Recommendation X.224)

Checksum algorithms

I.1 *Symbols*

The following symbols are used:

$\left.\begin{array}{l} C0 \\ C1 \end{array}\right\}$ variables used in the algorithms

i number (i.e., position) of an octet within the TPDU (see § 13.2)

n number (i.e., position) of the first octet of the checksum parameter

L length of the complete TPDU

X value of the first octet of the checksum parameter

Y value of the second octet of the checksum parameter.

I.2 *Arithmetic conventions*

Addition is performed in one of the two following modes:

a) modulo 255 arithmetic;

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

I.3 *Algorithm for generating checksum parameters*

I.3.1 Set up the complete TPDU with the value of the checksum parameter field set to zero.

I.3.2 Initialize $C0$ and $C1$ to zero.

I.3.3 Process each octet sequentially from $i = 1$ to L by

a) adding the value of the octet to $C0$; then

b) adding the value of $C0$ to $C1$.

I.3.4 Calculate X and Y such that

$$X = -C1 + (L - n) \cdot C0$$

$$Y = C1 - (L - n + 1) \cdot C0.$$

I.3.5 Place the values X and Y in octets n and $(n + 1)$ respectively.

Note – This algorithm calculates:

$$C1 = \sum_{i=1}^{L} (L - i + 1)a_i$$

which is equal to zero, if the formulas in § 6.17.3 are followed, since:

$$\sum_{i=1}^{L} (L - i + 1)a_i = (L + 1) \sum_{i=1}^{L} a_i - \sum_{i=1}^{L} ia_i = 0$$

I.4 *Algorithm for checking checksum parameters*

I.4.1 Initialize $C0$ and $C1$ to zero.

I.4.2 Process each octet of the TPDU sequentially from $i = 1$ to L by

a) adding the value of the octet to $C0$; then

b) adding the value of $C0$ to $C1$.

I.4.3 If, when all the octets have been processed, either or both of $C0$ and $C1$ does not have the value zero, the checksum formulas in § 6.17 have not been satisfied.

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

APPENDIX II

(to Recommendation X.224)

Differences between Recommendation X.224 and ISO 8073 (1986)

Recommendation X.224 and ISO 8073 (1986) are technically aligned except for the differences listed below:

II.1 *Defect reports*

In collaboration, ISO and CCITT have agreed to corrections resulting from defect reports. Recommendation X.224 includes the necessary corrections from reports 6, 11, 32, 33, 35, 65, 66, 67, 69, 71, 72, 73, 74, 77 (item 2), 80, 81, 82, 84, 85, 88, 89, 90, 91, 92, 93, 94, 95, 103, 106, and 107 which are not included in ISO 8073 (see Notes 1 and 2). An update of ISO 8073 has not yet been published (see Note 3).

Note 1 – Defect reports 1, 2, 3, 4, 5, 7, 9, 10, 12, 13, 14, 15, 16, 17, 19, 20, 21, 22, 23, 25, 26, 27, 28, 29 (items 1-7), 34, 36, 38, 39 (item 2), 40, 41, 42, 43, 45, 51, 52, 56, 57, 58, 59, 60, and 61 have had the necessaary changes applied to Recommendation X.224 and ISO 8073.

Note 2 – Defect report 18 does not apply to Recommendation X.224 or ISO 8073 and in addition defect reports 8, 24, 29 (item 8), 39 (item 1), 46, 48, 55, 62, 64 (item 2), 68, 76 (item 2), 77 (item 1), 79, 86, 87, 97, 99, and 104 have been rejected. No further changes will be made as a result of these defect reports.

Note 3 – ISO plans to issue a new edition of ISO 8073 shortly. This new edition of ISO 8073 is intended to be aligned with this Recommendation with respect to this difference.

II.2 *Conformance*

Recommendation X.224 in § 14, requires all systems to implement Class 0. ISO 8073, in Clause 14, requires all systems to implement either Class 0 or Class 2.

II.3 *Class negotiation*

An additional restriction regarding the classes of transport protocol that are proposed in a CR TPDU is defined in item (a) in § 14.4 of Recommendation X.224. No such restriction is contained in ISO 8073.

II.4 *Precedence*

ISO 8073, in its Annex A (state tables), includes the statement: «In the event of a discrepancy between the description in these tables and that contained in the text, the text takes precedence.» Recommendation X.224 does not contain this statement.

II.5 *Assignment to network connections*

In § 6.1.3 of Recommendation X.224 it is stated that the responder to the transport connect request becomes aware of the assignment when it receives particular TPDUs. In Clause 6.1.3 of ISO 8073 it is the non-owner of the network connection that becomes aware of the assignment upon receipt of these same TPDUs.

II.6 *Transport protocol identification*

The material contained in Annex B of Recommendation X.224 is not present in ISO 8073.

II.7 *Checksum algorithms*

The material contained in Appendix I to Recommendation X.224 is contained in Annex B to ISO 8073.

II.8 The material in this appendix is not present in ISO 8073 (see Note 3 above.)

Recommendation X.225

SESSION PROTOCOL SPECIFICATION FOR OPEN SYSTEMS INTERCONNECTION FOR CCITT APPLICATIONS[1]

(Malaga-Torremolinos, 1984; amended at Melbourne, 1988)

The CCITT,

considering

(a) that Recommendation X.200 defines the Reference Model of Open Systems Interconnection for CCITT Applications;

(b) that Recommendation X.215 specifies the Session Service Definition for Open Systems Interconnection for CCITT Applications;

(c) that Recommendation T.62 defines the Control Procedures for Teletex and Group 4 facsimile services,

unanimously declares

that this Recommendation defines the Session Protocol of Open Systems Interconnection for CCITT Applications as given in the Scope and Field of Application.

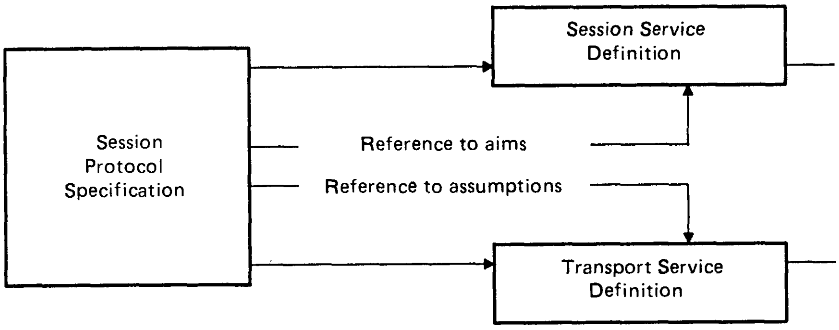
CONTENTS

[1] Recommendation X.225 is technically aligned with ISO 8327 [Information Processing Systems – Open Systems Interconnection – Basic Connection oriented session protocol specification] which includes corrections resulting from ISO defect reports numbered 8326/6, 8326/7, 8326/20, 8327/1, 8327/3, 8327/4 through 8327/10, 8327/12, 8327/17, 8327/18, 8327/19, 8327/26, 8327/27, 8327/30, 8327/34 and 8327/35 and Addendum 2 to incorporate unlimited user data, with the exception of the differences noted in Appendix I.

0 Introduction

The Session Protocol Standard is one of a set of Recommendations produced to facilitate the interconnection of computer systems. The set of Recommendations covers the services and protocols required to achieve such interconnection.

The Session Protocol Standard is positioned with respect to other related Recommendations by the layers defined in the Reference Model for Open Systems Interconnection (Recommendation X.200). It is most closely related to and lies within the field of application of the Session Service Definition (Recommendation X.215). It also uses and references the Transport Service Definition (Recommendation X.214), whose provisions it assumes in order to accomplish the aims of the session protocol. The interrelationship of these Recommendations is depicted in Figure 1/X.225.


Session Service Definition
Session Protocol Specification
Reference to aims
Reference to assumptions
Transport Service Definition

T0706630-88

FIGURE 1/X.225

Relationship between the session protocol and adjacent services

This Recommendation specifies a single protocol with a common encoding.

It is intended that the session protocol should be general enough to cater for the total range of session service users without restricting future extensions.

The protocol is structured so that subsets of protocol can be defined.

The primary aim of this Recommendation is to provide a set of rules for communication expressed in terms of the procedures to be carried out by peer session entities at the time of communication. These rules for communication are intended to provide a sound basis for development in order to serve a variety of purposes:

a) as a guide for implementors and designers;

b) for use in the testing and procurement of equipment;

c) as a part of an agreement for the admittance of systems into the open systems environment;

d) as a refinement to the understanding of OSI.

It is expected that the initial users of the Recommendation will be designers and implementors of equipment and the Recommendation contains, in Notes or in Annexes, guidance on the implementation of the procedures defined in the Recommendation.

It should be noted that, as the number of valid protocol sequences is very large, it is not possible with current technology to verify that an implementation will operate the protocol defined in this Recommendation correctly under all circumstances. It is possible by means of testing to establish confidence that an implementation correctly operates the protocol in a representative sample of circumstances. It is, however, intended that this Recommendation can be used in circumstances where two implementations fail to communicate in order to determine whether one or both have failed to operate the protocol correctly.

The variations and options available within this Recommendation are essential to enable a session service to be provided for a wide variety of applications. Thus, a minimally conforming implementation will not be suitable for use in all possible circumstances. It is important, therefore, to qualify all references to this Recommendation with statements of the options provided or required of with statements of the intended purpose of provision or use.

This Recommendation contains the following annexes and appendix:

a) Annex A – State Tables.

b) Annex B – Relationship to CCITT Recommendation T.62 encoding.

c) Annex C – PGIs and PIs reserved for use by Recommendation T.62.

d) Annex D – Compatibility between Protocol Version 1 and Protocol Version 2.

e) Appendix I – Difference between Recommendation X.225 and ISO International Standard 8327.

1 Scope and field of application

1.1 This Recommendation specifies:

a) procedures for a single protocol for the transfer of data and control information from one session entity to a peer session entity;

b) the means of selecting the functional units to be used by the session entities;

c) the structure and encoding of the session protocol data units used for the transfer of data and control information.

1.2 The procedures are defined in terms of:

a) the interactions between peer session entities through the exchange of session protocol data units;

b) the interactions between a session entity and the session service user in the same system through the exchange of session service primitives;

c) the interactions between a session entity and the transport service provider through the exchange of transport service primitives.

1.3 These procedures are applicable to instances of communication between systems which support the session layer of the OSI Reference Model and which wish to interconnect in an open systems environment.

1.4 This Recommendation also specifies conformance requirements for systems implementing these procedures. It does not contain tests which can be used to demonstrate this conformance.

2 References

Recommendation X.200 – Reference Model of Open Systems Interconnection for CCITT applications. (See also ISO 7498-1)

Recommendation X.214 – Transport Service Definition for Open Systems Interconnection for CCITT applications. (See also ISO 8072)

Recommendation X.215 – Session Service Definition for Open Systems Interconnection for CCITT applications. (See also ISO 8326 and ISO 8326 Addendum 2)

Recommendation T.62 – CCITT Recommendation T.62 – Control Procedures for the Teletex and Group 4 Facsimile Services.

ISO 7498-3 – Information processing systems – Open Systems Interconnection – Basic Reference Model – Part 3: Naming and Addressing[2].

Note – CCITT Recommendation T.62 is not essential for the application of this Recommendation, but is included in the list of references as it has been referred to, for information, in relation to interworking with the CCITT Telematic services (see Annexes B and C).

3 Definitions

Note – The definitions contained in this section make use of abbreviations defined in § 4.

3.1 This Recommendation is based on the concepts developed in Recommendation X.200, and makes use of the following terms defined in that Recommendation:

a) expedited session service data unit;

b) session connection;

c) session layer;

d) session protocol data unit;

e) session service;

f) session service access point;

g) session service data unit;

h) transport layer;

i) transport connection;

j) transport service;

k) transport service access point;

l) concatenation;

m) segmenting;

n) session selection (defined in ISO 7498-3).

3.2 This Recommendation is also based on concepts developed in Recommendation X.215 and makes use of the following terms defined in that Recommendation:

a) token;

b) calling SS-user;

c) called SS-user;

d) sending SS-user;

e) receiving SS-user;

f) requesting SS-user;

g) accepting SS-user;

h) requestor;

i) acceptor.

Note – The following terms used in this Recommendation are used in relation to tokens and are explained in Recommendation X.215:

a) assigned;

b) not assigned;

c) available;

d) not available.

[2] At present at the stage of draft; publication anticipated in due course.

3.3 For the purposes of this Recommendation, the following definitions apply:

3.3.1 **Session Protocol Machine; SPM**

An abstract machine that carries out the procedures specified in this protocol.

Note – A session entity is comprised of one or more SPMs.

3.3.2 **session service user; SS-user**

An abstract representation of the totality of those entities within a single system that make use of the Session Service.

3.3.3 **transport service provider; TS-provider**

An abstract machine which models the totality of the entities providing the transport service, as viewed by a session entity.

3.3.4 **local matter**

A decision made by a system concerning its behaviour in the Session Layer that is not subject to the requirements of this protocol.

3.3.5 **initiator**

An SPM that initiates a CONNECT SPDU.

3.3.6 **responder**

An SPM with whom an initiator wishes to establish a session connection.

Note – Initiator and responder are defined with respect to a single session connection.

3.3.7 **sending SPM**

An SPM that sends a given SPDU.

3.3.8 **receiving SPM**

An SPM that receives a given SPDU.

3.3.9 **owner (of a token)**

The SPM to whom a token is assigned.

3.3.10 **proposed parameter**

The value for a parameter proposed by an SPM in a CONNECT SPDU or an ACCEPT SPDU that it wishes to use on the session connection.

3.3.11 **negotiation**

The process by which two SPMs agree on a common set of functional units and protocol values and on the initial setting of available tokens.

3.3.12 **selected parameter**

The value for a parameter that has been chosen for use on the session connection.

3.3.13 **valid SPDU**

An SPDU which complies with the requirements of this Recommendation with respect to structure and encoding.

3.3.14 **invalid SPDU**

An SPDU which does not comply with the requirements of this Recommendation with respect to structure and encoding.

3.3.15 **protocol error**

Use of an SPDU that does not comply with the procedures agreed for the session connection.

3.3.16 **transparent (data)**

SS-user data which is transferred intact between SPMs and which is unavailable for use by the SPMs.

3.3.17 **SPDU identifier (SI)**

Heading information that identifies the SPDU concerned.

3.3.18 **length indicator (LI)**

An indicator that represents the length of an associated parameter field.

3.3.19 **parameter field**

A group of one or more octets used to represent a particular set of information.

3.3.20 **parameter identifier (PI)**

An identifier, defined in this Recommendation, that indicates the type of information contained in its associated parameter field.

3.3.21 **PI unit**

An element of an SPDU that contains a PI field together with its associated LI field and parameter field.

3.3.22 **parameter group identifier (PGI)**

An identifier, defined in this Recommendation, that indicates the type of information contained in its associated parameter field. The associated parameter field may consist of a set of PI units.

3.3.23 **PGI unit**

An element of an SPDU that contains a PGI field together with its associated LI field and parameter field.

3.3.24 **parameter value (PV)**

Information that represents the value of the parameter identified by either a PI or PGI.

3.3.25 **local variable**

A local variable within the SPM which is used as a means of clarifying the effects of certain actions and clarifying the conditions under which certain actions are permitted.

4 Symbols and abbreviations

4.1 *Data units*

SPDU	Session Protocol Data Unit
SSDU	Session Service Data Unit
TSDU	Transport Service Data Unit

4.2 *SPDU fields*

SI	SPDU Identifier (see § 3.3.17)
LI	Length Indicator (see § 3.3.18)
PI	Parameter Identifier (see § 3.3.20)
PGI	Parameter Group Identifier (see § 3.3.22)
PV	Parameter Value (see § 3.3.24)

4.3 *Timer variables*

TIM	Disconnection and abort timer

4.4 *Miscellaneous*

SPM	Session Protocol Machine (see § 3.3.1)
SS	Session Service
SSAP	Session service access point
TSAP	Transport service access point

4.5 *Local variables*

Vact	See § 5.8.1
Vnextact	See § 5.8.2
V(A)	See § 5.8.3
V(M)	See § 5.8.4
V(R)	See § 5.8.5
Vsc	See § 5.8.6

5 **Overview of the session protocol**

5.1 *Model of the session layer*

The SPM (see Note) within the session layer communicates with the SS-user through an SSAP by means of the service primitives as defined by the session service definition in Recommendation X.215. Service primitives will cause or be the result of session protocol data unit exchanges between the peer SPMs using a transport connection. These protocol exchanges are effected using the services of the transport layer as defined by the transport service definition in Recommendation X.214 through two TSAPs.

Session connection endpoints are identified in end systems by an internal, implementation dependent mechanism, so that the SS-user and the SPM can refer to each session connection.

The model of the Session Layer is illustrated in Figure 2/X.225.

Note – A session entity is comprised of one or more SPMs.

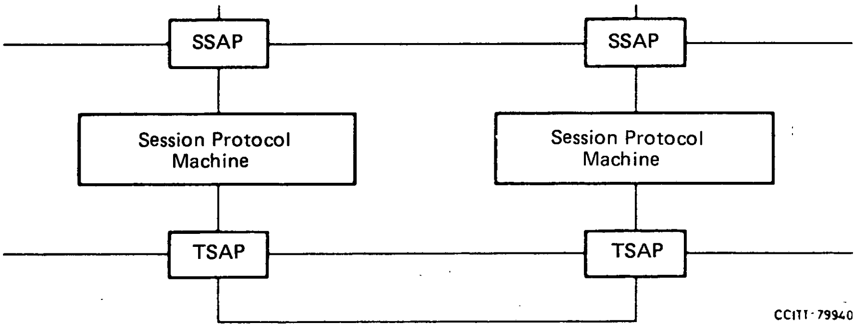


FIGURE 2/X.225

Model of the session layer

5.2 *Services provided by the session layer*

The protocol specified in this Recommendation supports the session service defined in Recommendation X.215. Information is transferred to and from the SS-user using the session service primitives listed in Table 1/X.225. Table 1/X.225 also defines the SPDUs associated with each of the service primitives.

TABLE 1/X.225

Session service primitives

Service	Primitives	Associated SPDUs
Session Connection	S-CONNECT request S-CONNECT indication S-CONNECT (accept) response S-CONNECT (accept) confirm S-CONNECT (reject) response S-CONNECT (reject) confirm	CONNECT SPDU CONNECT SPDU ACCEPT SPDU ACCEPT SPDU REFUSE SPDU REFUSE SPDU
Normal Data Transfer	S-DATA request S-DATA indication	DATA TRANSFER SPDU DATA TRANSFER SPDU
Expedited Data Transfer	S-EXPEDITED-DATA request S-EXPEDITED-DATA indication	EXPEDITED DATA SPDU EXPEDITED DATA SPDU
Typed Data Transfer	S-TYPED-DATA request S-TYPED-DATA indication	TYPED DATA SPDU TYPED DATA SPDU
Capability Data Exchange	S-CAPABILITY-DATA request S-CAPABILITY-DATA indication S-CAPABILITY-DATA response S-CAPABILITY-DATA confirm	CAPABILITY DATA SPDU CAPABILITY DATA SPDU CAPABILITY DATA ACK SPDU CAPABILITY DATA ACK SPDU
Give Tokens	S-TOKEN-GIVE request S-TOKEN-GIVE indication	GIVE TOKENS SPDU GIVE TOKENS SPDU
Please Tokens	S-TOKEN-PLEASE request S-TOKEN-PLEASE indication	PLEASE TOKENS SPDU PLEASE TOKENS SPDU
Give Control	S-CONTROL-GIVE request S-CONTROL-GIVE indication	GIVE TOKENS CONFIRM SPDU GIVE TOKENS CONFIRM SPDU

TABLE 1/X.225 *(cont.)*

Service	Primitives	Associated SPDUs
Minor Synchronization Point	S-SYNC-MINOR request S-SYNC-MINOR indication S-SYNC-MINOR response S-SYNC-MINOR confirm	MINOR SYNC POINT SPDU MINOR SYNC POINT SPDU MINOR SYNC ACK SPDU MINOR SYNC ACK SPDU
Major Synchronization Point	S-SYNC-MAJOR request S-SYNC-MAJOR indication S-SYNC-MAJOR response S-SYNC-MAJOR confirm	MAJOR SYNC POINT SPDU MAJOR SYNC POINT SPDU MAJOR SYNC ACK SPDU MAJOR SYNC ACK SPDU
Resynchronize	S-RESYNCHRONIZE request S-RESYNCHRONIZE indication S-RESYNCHRONIZE response S-RESYNCHRONIZE confirm	RESYNCHRONIZE SPDU RESYNCHRONIZE SPDU RESYNCHRONIZE ACK SPDU RESYNCHRONIZE ACK SPDU
P-Exception Report	S-P-EXCEPTION-REPORT indication	EXCEPTION REPORT SPDU
U-Exception Reporting	S-U-EXCEPTION-REPORT request S-U-EXCEPTION-REPORT indication	EXCEPTION DATA SPDU EXCEPTION DATA SPDU
Activity Start	S-ACTIVITY-START request S-ACTIVITY-START indication	ACTIVITY START SPDU ACTIVITY START SPDU
Activity Resume	S-ACTIVITY-RESUME request S-ACTIVITY-RESUME indication	ACTIVITY RESUME SPDU ACTIVITY RESUME SPDU

TABLE 1/X.225 *(end)*

Service	Primitives	Associated SPDUs
Activity Interrupt	S-ACTIVITY-INTERRUPT request S-ACTIVITY-INTERRUPT indication S-ACTIVITY-INTERRUPT response S-ACTIVITY-INTERRUPT confirm	ACTIVITY INTERRUPT SPDU ACTIVITY INTERRUPT SPDU ACTIVITY INTERRUPT ACK SPDU ACTIVITY INTERRUPT ACK SPDU
Activity Discard	S-ACTIVITY-DISCARD request S-ACTIVITY-DISCARD indication S-ACTIVITY-DISCARD response S-ACTIVITY-DISCARD confirm	ACTIVITY DISCARD SPDU ACTIVITY DISCARD SPDU ACTIVITY DISCARD ACK SPDU ACTIVITY DISCARD ACK SPDU
Activity End	S-ACTIVITY-END request S-ACTIVITY-END indication S-ACTIVITY-END response S-ACTIVITY-END confirm	ACTIVITY END SPDU ACTIVITY END SPDU ACTIVITY END ACK SPDU ACTIVITY END ACK SPDU
Orderly Release	S-RELEASE request S-RELEASE indication S-RELEASE (accept) response S-RELEASE (accept) confirm S-RELEASE (reject) response S-RELEASE (reject) confirm	FINISH SPDU FINISH SPDU DISCONNECT SPDU DISCONNECT SPDU NOT FINISHED SPDU NOT FINISHED SPDU
U-Abort	S-U-ABORT request S-U-ABORT indication	ABORT SPDU ABORT SPDU
P-Abort	S-P-ABORT indication	ABORT SPDU

5.3 *Services assumed from the transport layer*

The protocol specified in this Recommendation assumes the use of the connection-oriented transport service defined in Recommendation X.214.

Information is transferred to and from the TS provider in the transport service primitives listed in Table 2/X.225.

TABLE 2/X.225

Transport service primitives

Primitive	X/Y	Parameters
T-CONNECT request indication	X	Called Address, Calling Address, Expedited Data option, Quality of Service, TS User-Data
T-CONNECT response confirm	X	Quality of Service, Responding Address, Expedited Data option, TS User-Data
T-DATA request indication	X	TS-User-Data
T-EXPEDITED-DATA request indication	Y	TS User-Data
T-DISCONNECT request	X	TS User-Data
T-DISCONNECT indication	X	Disconnect reason, TS User-Data

X: The session protocol assumes that this service is always available.

Y: The session protocol assumes that this service is provided by the transport layer when requested by the SPM during the session connection establishment phase.

5.4 *Functions of the session layer*

5.4.1 *Overview of functions*

The functions in the session layer are those necessary to bridge the gap between the services available from the transport layer and those offered to the SS-users.

The functions in the session layer are concerned with dialogue management, data flow synchronization, and data flow resynchronization. They are described below; the descriptions are grouped into those concerned with the connection establishment phase, the data transfer phase, and the release phase.

5.4.2 *Connection establishment phase*

The purpose of the connection establishment phase is to establish a session connection between two SS-users, and:

a) to map session addresses onto transport addresses;

b) to select transport quality of service parameters needed (see § 6.1.4);

c) to negotiate session parameters (see §§ 7.1, 7.2 and 7.4);

d) to transfer session selectors (see §§ 7.1 and 7.4) if required;

e) to distinguish between session connections (see §§ 7.1 and 7.4);

f) to transfer transparent user data (see §§ 7.1, 7.2, 7.3 and 7.4);

g) to select a protocol version (see Note).

Note – This Recommendation specifies the following protocol versions:

i) Protocol Version 1 which imposes restrictions on the length of the user data field;

ii) Protocol Version 2 which imposes no explicit restrictions on the length of the user data field.

Annex D identifies the compatibility between Protocol Version 1 and Protocol Version 2.

5.4.3 *Data transfer phase*

The purpose of the data transfer phase is to transport SSDUs between two SS-users connected by a session connection. This purpose is achieved by means of transmission of SPDUs and by the following functions, each of which may or may not be used, depending on the functional units selected in the session connection establishment phase. These concepts are defined in Recommendation X.215:

a) *Normal Data Transfer* (see § 7.11), which may involve segmenting of SSDUs into SPDUs and reassembly by the destination SPM; and concatenation and separation of certain SPDUs. There are two modes of operation:

1) Half-duplex, when the right to send data is restricted to the owner of the data token;

2) Duplex, when there is no restriction on the right to send data.

b) *Token management* (see §§ 7.16 to 7.19), to enable the SS-users to request and transfer tokens which control the exclusive right to exercise certain functions (see Table 5/X.225).

c) *Exception Reporting* (see §§ 7.27 and 7.28), to enable the SS-provider or the SS-user to report exception conditions that are less severe than those requiring abort.

d) *Typed Data Transfer* (see § 7.13), to enable transfer of information which is not subject to assignment of the data token.

e) *Minor Synchronization Point* (see §§ 7.20 and 7.21), to enable the SS-users to define minor synchronization points in the normal data flow. These minor synchronization points may optionally be confirmed, but have no implications on the data flow. Minor synchronization points are identified by synchronization point serial numbers. The serial number is incremented by one on each occasion that a minor synchronization point is placed in the data flow, and each time a minor synchronization point is received, such that both SS-users have the same serial numbers for the same synchronization point.

f) *Major Synchronization Point* (see §§ 7.22 and 7.23 and e) above), to enable the SS-users to define major synchronization points in the normal data flow. These major synchronization points are required to be confirmed before the requesting SS-user is permitted to send any subsequent data on either the normal flow or the expedited flow and as such clearly separate the dialogue units.

g) *Resynchronize* (see §§ 7.24 and 7.25), a function that allows a session connection to be set or reset to a defined synchronization point and reassign the tokens.

h) *Expedited Data Transfer* (see § 7.12), a function used to convey a limited amount of user data with special handling. Such data may bypass normal data en route, but will be delivered prior to any data subsequently sent on the transport normal flow or the transport expedited flow.

i) *Activity Management* (see §§ 7.29 to 7.36) provides a means explicitly to start, end, resume, interrupt or discard an activity. This provides a way:

   1) to identify the entered activity and commence synchronization point serial numbering;

   2) to identify the continued activity and reset the synchronization point serial number in case of resumption.

j) *Capability Data Exchange* (see §§ 7.14 and 7.15), to provide a confirmed transfer of a limited amount of user data.

5.4.4 *Connection release phase*

The purpose of the release phase is to provide disconnection of the session connection, by using the following functions:

a) orderly release (negotiated and non-negotiated);

b) abort (provider and user initiated);

c) transfer of transparent user data.

5.5 *Functional units*

Functional units are logical groupings of related elements of procedure defined by this Recommendation for the purpose of:

a) negotiation for use during session connection establishment;

b) specification of conformance requirements.

The SPDUs associated with elements of procedure for each functional unit are specified in Table 3/X.225.

Tokens are associated with functional units (see § 5.6).

5.5.1 *Kernel functional unit*

The kernel functional unit supports the basic protocol elements of procedure required to establish a session connection, transfer normal data and release the session connection.

5.5.2 *Negotiated release functional unit*

The negotiated release functional unit supports the negotiated release service which enables the SS-users to negotiate the orderly release of the session connection. If this functional unit has been selected, an attempt to release the session connection may be refused by the accepting SS-user.

5.5.3 *Half-duplex functional unit*

The half-duplex functional unit is used to control the right to send data. It is not valid to select both this functional unit and the duplex functional unit for use on the same session connection.

5.5.4 *Duplex functional unit*

The duplex functional unit is used when the right to send data is not controlled. It is not valid to select both this functional unit and the half-duplex functional unit for use on the same session connection.

5.5.5 *Expedited data functional unit*

The expedited data functional unit supports the expedited data service and allows the transfer of a limited amount of SS-user data.

The services supported by this functional unit can only be requested when the transport expedited flow is available to this session connection.

5.5.6 *Typed data functional unit*

The typed data functional unit enables the SS-users to transfer data in a manner which is not subject to the control imposed by the availability of the data token.

TABLE 3/X.225

Functional units

Functional unit	SPDU code	SPDU name	Reference
Kernel	CN OA CDO AC RF FN DN AB AA DT	CONNECT (Note 1) OVERFLOW ACCEPT (Note 2) CONNECT DATA OVERFLOW (Note 2) ACCEPT (Note 1) REFUSE (Note 1) FINISH DISCONNECT ABORT ABORT ACCEPT (Note 3) DATA TRANSFER	7.1 7.2 7.3 7.4 7.5 7.6 7.7 7.9 7.10 7.11
Negotiated release	NF GT PT	NOT FINISHED GIVE TOKENS (Note 5) PLEASE TOKENS (Note 5)	7.8 7.16 7.17
Half-duplex	GT PT	GIVE TOKENS (Note 4) PLEASE TOKENS (Note 4)	7.16 7.17
Duplex		No additional associated SPDUs	
Expedited data	EX	EXPEDITED DATA	7.12
Typed data	TD	TYPED DATA	7.13
Capability data exchange	CD CDA	CAPABILITY DATA CAPABILITY DATA ACK	7.14 7.15
Minor synchronize	MIP MIA GT PT	MINOR SYNC POINT MINOR SYNC ACK GIVE TOKENS (Note 6) PLEASE TOKENS (Note 6)	7.20 7.21 7.16 7.17
Major synchronize	MAP MAA PR GT PT	MAJOR SYNC POINT MAJOR SYNC ACK PREPARE (Note 7) GIVE TOKENS (Note 8) PLEASE TOKENS (Note 8)	7.22 7.23 7.26 7.16 7.17
Resynchronize	RS RA PR	RESYNCHRONIZE RESYNCHRONIZE ACK PREPARE (Note 7)	7.24 7.25 7.26
Exceptions	ER ED	EXCEPTION REPORT EXCEPTION DATA	7.27 7.28

TABLE 3/X.225 *(cont.)*

Functional unit	SPDU code	SPDU name	Reference
Activity management	AS	ACTIVITY START	7.29
	AR	ACTIVITY RESUME	7.30
	AI	ACTIVITY INTERRUPT	7.31
	AIA	ACTIVITY INTERRUPT ACK	7.32
	AD	ACTIVITY DISCARD	7.33
	ADA	ACTIVITY DISCARD ACK	7.34
	AE	ACTIVITY END	7.35
	AEA	ACTIVITY END ACK	7.36
	PR	PREPARE (Note 7)	7.26
	GT	GIVE TOKENS (Note 8)	7.16
	PT	PLEASE TOKENS (Note 8)	7.17
	GTC	GIVE TOKENS CONFIRM (Note 9)	7.18
	GTA	GIVE TOKENS ACK (Note 9)	7.19

Note 1 – An implementation (see § 9) is required to be able to:

a) send a CONNECT SPDU and receive an ACCEPT SPDU or a REFUSE SPDU, or

b) receive a CONNECT SPDU and send an ACCEPT SPDU or a REFUSE SPDU, or

c) send and receive both.

Note 2 – These SPDUs are only used when the SSDU passed in the S-CONNECT request is segmented [see § 6.3.5 b)].

Note 3 – Reception and correct action is mandatory; transmission is optional if the transport connection is not to be reused (see § 7.10.2).

Note 4 – Used to manage the data token.

Note 5 – Used to manage the release token.

Note 6 – Used to manage the synchronize-minor token.

Note 7 – PREPARE SPDU is mandatory if the transport expedited flow is available to this session connection, otherwise it is not used (see § 6.4).

Note 8 – Used to manage the major/activity token.

Note 9 – Used only on session connections on which activity management has been selected, for giving all available tokens, when no activity is in progress.

5.5.7 *Capability data exchange functional unit*

The capability data functional unit supports the capability data exchange service, which allows a confirmed transfer of SS-user data when the activity management functional unit has been selected, but when no activity is in progress.

5.5.8 *Minor synchronize functional unit*

The minor synchronize functional unit supports the minor synchronization service which enables the SS-user to request that the SPM places minor synchronization points in the normal data flow. These minor synchronization points are identified by serial numbers.

5.5.9 *Major synchronize functional unit*

The major synchronize functional unit supports the major synchronize service which enables the SS-user to request that the SPM places major synchronization points in the normal data flow. These major synchronization before ane identified by serial numbers, and clearly separate the data flow before after the major synchronization point.

5.5.10 *Resynchronize functional unit*

The resynchronize functional unit supports the resynchronize service which enables the SS-users to modify the synchronization point serial number and reassign the tokens.

5.5.11 *Exceptions functional unit*

The exceptions functional unit allows both the SPM and the SS-users to report detected errors, rather than aborting the session connection.

This functional unit can only be selected when the half-duplex functional unit has been selected.

5.5.12 *Activity management functional unit*

The activity management functional unit supports the activity management services which allow the SS-users to manage synchronized logical pieces of work.

5.6 *Tokens*

Table 4/X.225 specifies those functional units that have tokens associated with them.

The SPM may only send an SPDU listed in Table 5/X.225 (and accept the associated service primitive) subject to the availability and assignment of tokens defined in that table.

TABLE 4/X.225

Tokens associated with functional units

Functional unit	Token
Negotiated release	release token
Half-duplex	data token
Minor synchronize	synchronize-minor token
Major synchronize	major/activity token
Activity management	major/activity token

TABLE 5/X.225

Token restrictions

SPDUs	data token	synchronize-minor token	major/activity token	release token
FINISH SPDU	2	2	2	2
NOT FINISHED SPDU	nr	nr	nr	0
DATA TRANSFER SPDU (Half-duplex)	1	nr	nr	nr
DATA TRANSFER SPDU (Duplex)	3	nr	nr	nr
CAPABILITY DATA SPDU	2	2	1	nr
GIVE TOKEN SPDU:				
Data token	1	nr	nr	nr
synchronize-minor token	nr	1	nr	nr
major/activity token	nr	nr	1	nr
release token	nr	nr	nr	1
PLEASE TOKEN SPDU:				
data token	0	nr	nr	nr
synchronize-minor token	nr	0	nr	nr
major/activity token	nr	nr	0	nr
release token	nr	nr	nr	0
GIVE TOKENS CONFIRM SPDU	2	2	1	2
MINOR SYNC POINT SPDU	2	1	nr	nr
MAJOR SYNC POINT SPDU	2	2	1	nr
EXCEPTION REPORT SPDU	0	nr	nr	nr
EXCEPTION DATA SPDU	0	nr	nr	nr
ACTIVITY START SPDU	2	2	1	nr
ACTIVITY RESUME SPDU	2	2	1	nr
ACTIVITY INTERRUPT SPDU	nr	nr	1	nr
ACTIVITY DISCARD SPDU	nr	nr	1	nr
ACTIVITY DISCARD SPDU	2	2	1	nr

0: Token available and not assigned to the SS-user who initiated the associated service primitive
1: Token available and assigned to the SS-user who initiated the associated service primitive
2: Token not available or token assigned to the SS-user who initiated the associated service primitive
3: Token not available
nr: No restriction

5.7 *Negotiation*

Negotiation takes place between both SPMs during session connection establishment according to the following rules.

5.7.1 *Negotiation of functional units*

Each SPM proposes use or non-use of each functional unit, except for the kernel functional unit, based on requirements from the SS-users. The functional unit is selected only if both the initiator and the responder propose use of the functional unit.

The capability data exchange functional unit can only be proposed if the activity management functional unit is also proposed.

The exceptions functional unit can only be proposed if the half-duplex functional unit is also proposed.

5.7.2 *Negotiation of initial token settings*

When the initiator proposes use of a functional unit that requires a token, it also proposes the initial token setting:

a) initiator's side;

b) responder's side;

c) called SS-user's choice.

If use of the functional unit is selected, the token is set to the side proposed by the initiator. If the initiator proposed "called SS-user choice", the responder's proposed token setting is selected.

5.7.3 *Negotiation of initial serial number*

When the initiator proposes any of the minor synchronize, major synchronize or resynchronize functional units but does not propose the activity management functional unit, it also proposes an initial serial number.

When the initiator proposes any of the minor synchronize, major synchronize or resynchronize functional units and also proposes the activity management functional unit, it may also propose an initial serial number.

In all other cases, the initiator does not propose an initial serial number.

When the responder proposes any of the minor synchronize, major synchronize or resynchronize functional units but does not propose the activity management functional unit, it also proposes an initial serial number, which is the first serial number to be used.

In all other cases, the responder does not propose an initial serial number.

5.7.4 *Negotiation of version number*

Each SPM indicates all the appropriate versions of the protocol that it is capable of supporting.

5.7.5 *Negotiation of maximum TSDU size*

Each SPM proposes a maximum TSDU size that the initiator is permitted to send. The lesser of the two numbers is used. A zero value is interpreted to mean unlimited TSDU size. If either SPM proposes zero, the initiator may not send segmented data or typed data SSDUs.

Each SPM also proposes a maximum TSDU size that the responder is permitted to send. The lesser of the two numbers is used. A zero value is interpreted to mean unlimited TSDU size. If either SPM proposes zero, the responder may not send segmented data or typed data SSDUs on the session connection.

5.8 *Local variables*

This Recommendation uses local variables as a means of clarifying the effect of certain actions and clarifying the conditions under which certain actions are valid.

5.8.1 *Vact*

Vact is used by the SPM to determine if an activity is in progress when the activity management functional unit has been selected:

Vact = true: an activity is in progress;

Vact = false: no activity is in progress.

5.8.2 *Vnextact*

Vnextact is used by the SPM when the activity management functional unit has been selected:

Vnextact = true: a MAJOR SYNC POINT SPDU has been sent or received;

Vnextact = false: an ACTIVITY END SPDU has been sent or received.

5.8.3 *V(A)*

V(A) is used by the SPM and is the lowest serial number to which a synchronization point confirmation is expected. No confirmation is expected when V(A) = V(M).

5.8.4 *V(M)*

V(M) is used by the SPM and is the next serial number to be used.

5.8.5 *V(R)*

V(R) is used by the SPM and is the lowest serial number to which resynchronization restart is permitted.

5.8.6 *Vsc*

Vsc is used by the SPM to determine whether or not the SS-user has the right to send minor synchronization point responses. Vsc has the following values:

Vsc = true: the SS-user has the right to issue minor synchronization point responses when V(A) is less than V(M);

Vsc = false: the SS-user does not have the right to issue minor synchronization point responses.

Note – The manipulation of V(A), V(M), V(R) and Vsc and the circumstances under which they are updated are specified in § 7 and are summarized in a Table A-4/X.225 in Annex A.

6 Use of the transport service

This section defines the way that the transport service primitives are used by the SPM.

6.1 *Assignment of a session connection to the transport connection*

6.1.1 *Purpose*

Assignment of a session connection to a transport connection.

6.1.2 *Transport service primitives*

The procedure uses the following transport service primitives:

T-CONNECT request
T-CONNECT indication
T-CONNECT response
T-CONNECT confirm
T-DISCONNECT request
T-DISCONNECT indication

6.1.3 *SPDUs used*

No SPDUs are used during assignment to a transport connection.

6.1.4 *Description*

A session connection is assigned to an existing transport connection suitable for reuse, or a new transport connection is created for the purpose. This assignment is based on the quality of service (see Recommendation X.215) requested by the SS-user in the S-CONNECT request.

If a transport connection is established with the transport expedited data option, the transport expedited data flow is available for the duration of the transport connection. Use of transport expedited data is specified in § 6.4.

The transport expedited flow is requested by the SPM when the T-CONNECT request is issued if:

a) the SS-user requested the expedited data functional unit, or

b) the SS-user requested an extended control QOS for the session connection.

Only the initiator of the transport connection is permitted to issue the CONNECT SPDU.

When a session connection is terminated, the underlying transport connection is also terminated, unless reuse of the transport connection has been agreed.

Use of the TS-user data parameter in T-CONNECT request, indication, response and confirm is reserved for future use. When T-CONNECT request or a T-CONNECT response is issued, this parameter is empty. When a T-CONNECT indication or T-CONNECT confirm is received, this parameter is ignored.

6.2 *Reuse of the transport connection*

6.2.1 *Purpose*

To allow the transport connection to be retained for reuse by another session connection.

6.2.2 *Transport service primitives*

The procedure uses the following transport service primitives:

T-DATA request
T-DATA indication

6.2.3 *SPDUs used*

The following SPDUs are related to reuse of the transport connection:

REFUSE SPDU (see § 7.5);
FINISH SPDU (see § 7.6);
DISCONNECT SPDU (see § 7.7);
ABORT SPDU (see § 7.9);
ABORT ACCEPT SPDU (see § 7.10).

6.2.4 *Description*

When a session connection is refused, or has been successfully connected and subsequently disconnected, by abort or orderly release, the supporting transport connection may be either disconnected or reused.

The transport connection may be kept for reuse provided that the transport expedited flow is not available, and either:

a) the SPM which established the transport connection requests retention of the transport connection by parameter in an ABORT SPDU or a FINISH SPDU, or

b) the SPM which established the transport connection receives a REFUSE SPDU or an ABORT SPDU which indicates by parameter that the transport connection is to be retained.

To avoid contention for a retained transport connection, only the transport connection initiator may reuse the transport connection by sending a CONNECT SPDU to establish a new session connection.

6.3 *Use of transport normal data*

6.3.1 *Purpose*

To convey SPDUs in user data fields of transport service normal data primitives.

6.3.2 *Transport service primitives*

The procedure uses the following transport service primitives:
T-DATA request
T-DATA indication

6.3.3 *SPDUs used*

The following SPDUs are sent on the transport normal flow:
CONNECT SPDU (see § 7.1);
OVERFLOW ACCEPT SPDU (see § 7.2);
CONNECT DATA OVERFLOW SPDU (see § 7.3);
ACCEPT SPDU (see § 7.4);
REFUSE SPDU (see § 7.5);
FINISH SPDU (see § 7.6);
DISCONNECT SPDU (see § 7.7);
NOT FINISHED SPDU (see § 7.8);
DATA TRANSFER SPDU (see § 7.11);
TYPED DATA SPDU (see § 7.13);
CAPABILITY DATA SPDU (see § 7.14);
CAPABILITY DATA ACK SPDU (see § 7.15);
GIVE TOKENS SPDU (see (§ 7.16);
PLEASE TOKENS SPDU (see § 7.17);
GIVE TOKENS CONFIRM SPDU (see § 7.18);
GIVE TOKENS ACK SPDU (see § 7.19);
MINOR SYNC POINT SPDU (see § 7.20);
MINOR SYNC ACK SPDU (see § 7.21);
MAJOR SYNC POINT SPDU (see § 7.22);
MAJOR SYNC ACK SPDU (see § 7.23);
RESYNCHRONIZE SPDU (see § 7.24);
RESYNCHRONIZE ACK SPDU (see § 7.25);
EXCEPTION REPORT SPDU (see § 7.27);
EXCEPTION DATA SPDU (see § 7.28);
ACTIVITY START SPDU (see § 7.29);
ACTIVITY RESUME SPDU (see § 7.30);
ACTIVITY INTERRUPT SPDU (see § 7.31);
ACTIVITY INTERRUPT ACK SPDU (see § 7.32);
ACTIVITY DISCARD SPDU (see § 7.33);
ACTIVITY DISCARD ACK SPDU (see § 7.34);
ACTIVITY END SPDU (see § 7.35);
ACTIVITY END ACK SPDU (see § 7.36).

If the SS-user data exceeds 9 octets or if the transport extended flow is not available, the following additional SPDUs are sent on the transport normal flow:
ABORT SPDU (see § 7.9);
ABORT ACCEPT SPDU (see § 7.10).

6.3.4 *Transfer of SPDUs*

The SPDUs listed in § 6.3.3 are transferred using the transport normal data transfer service.

6.3.5 *Segmenting*

Segmenting of SSDUs takes place under the following circumstances:

a) when a maximum TSDU size has been selected, in which case a data SSDU or a typed data SSDU may be mapped on to more than one SPDU;

b) when protocol version 2 is proposed or selected and either

   i) the SPDU size would exceed the maximum TSDU size, or

   ii) the SPDU size would exceed 65539 octets for an SPDU to be sent on the transport normal flow or 16 octets for an SPDU to be sent on the transport expedited flow

in which case SSDUs other than data SSDUs, typed data SSDUs and expedited data SSDUs are mapped into more than one SPDU.

In all other cases, each SSDU is mapped one-to-one onto an SPDU.

Note – Implementors should note that when segmenting is selected:

a) the control information of each SPDU indicates whether or not it contains the first or last segment of the SSDU;

b) the size of the segments of the SSDU is constrained by the maximum TSDU size selected for that direction of transfer.

6.3.6 *Maximum TSDU size*

When a maximum TSDU size has been selected, the SPDU size may not exceed the maximum TSDU size selected for that direction of transfer and a sequence of concatenated SPDUs may not exceed the maximum TSDU size selected for that direction of transfer.

6.3.7 *Concatenation*

Each SPDU is defined in Table 6/X.225 as belonging to one of the following categories:

a) *Category 0 SPDUs* which may be mapped one-to-one onto a TSDU or may be concatenated with one or more Category 2 SPDUs;

b) *Cateogry 1 SPDUs* which are always mapped one-to-one onto a TSDU;

c) *Category 2 SPDUs* which are never mapped one-to-one onto a TSDU.

Basic concatenations of a category 0 SPDU with a single category 2 SPDU, defined as valid and in the order indicated in Table 7/X.225, may always be mapped onto a single TSDU.

If the receiving SPM has indicated that it can accept extended concatenation, the sending SPM may map a category 0 SPDU with one or more category 2 SPDUs (as specified in Table 8/X.225) onto a single TSDU in the case where this concatenated sequence does not fit into a single TSDU, extended concatenation cannot be applied.

The valid mappings of SPDUs onto TSDUs are illustrated in Figure 3/X.225.

Any other concatenation of SPDUs is defined as invalid.

6.3.7.1 *Processing order of concatenated SPDUs*

On receipt of SPDUs that have been concatenated using basic concatenation, the category 2 SPDUs are processed before the category 0 SPDU.

On receipt, SPDUs that have been concatenated using extended concatenation are processed in the following order:

a) ACTIVITY START SPDU or
ACTIVITY RESUME SPDU;

b) DATA TRANSFER SPDU;

c) MINOR SYNC POINT SPDU or
MINOR SYNC ACK SPDU or
MAJOR SYNC POINT SPDU or
MAJOR SYNC ACK SPDU or
ACTIVITY END SPDU or
ACTIVITY END ACK SPDU;

d) GIVE TOKENS SPDU or
PLEASE TOKENS SPDU.

TABLE 6/X.225

Category 0, 1 and 2 SPDUs

Category 0 SPDUs	Category 1 SPDUs	Category 2 SPDUs
GIVE TOKENS SPDU	CONNECT SPDU	DATA TRANSFER SPDU
PLEASE TOKENS SPDU	ACCEPT SPDU	
		MINOR SYNC POINT SPDU
	REFUSE SPDU	MINOR SYNC POINT SPDU
	FINISH SPDU	
	DISCONNECT SPDU	MAJOR SYNC POINT SPDU
	NOT FINISHED SPDU	MAJOR SYNC ACK SPDU
	ABORT SPDU	
	ABORT ACCEPT SPDU	RESYNCHRONIZE SPDU
		RESYNCHRONIZE ACK SPDU
	GIVE TOKENS CONFIRM SPDU	
	GIVE TOKENS ACK SPDU	ACTIVITY START SPDU
		ACTIVITY RESUME SPDU
	EXPEDITED SPDU	ACTIVITY DISCARD SPDU
	PREPARE SPDU	ACTIVITY DISCARD ACK SPDU
	TYPED DATA SPDU	ACTIVITY INTERRUPT SPDU
		ACTIVITY INTERRUPT ACK SPDU
		ACTIVITY END SPDU
	OVERFLOW ACCEPT SPDU	ACTIVITY END ACK SPDU
	CONNECT DATA OVERFLOW SPDU	CAPABILITY DATA SPDU
		CAPABILITY DATA ACK SPDU
		EXCEPTION REPORT SPDU
		EXCEPTION DATA SPDU

TABLE 7/X.225

Valid basic concatenation of SPDUs

First SPDU	Second SPDU
GIVE TOKENS SPDU	DATA TRANSFER SPDU
GIVE TOKENS SPDU PLEASE TOKENS SPDU	MINOR SYNC POINT SPDU MINOR SYNC ACK SPDU
GIVE TOKENS SPDU PLEASE TOKENS SPDU	MAJOR SYNC POINT SPDU MAJOR SYNC ACK SPDU
GIVE TOKENS SPDU [a)] PLEASE TOKENS SPDU	RESYNCHRONIZE SPDU RESYNCHRONIZE ACK SPDU
GIVE TOKENS SPDU GIVE TOKENS SPDU	ACTIVITY START SPDU ACTIVITY RESUME SPDU
GIVE TOKENS SPDU [a)] PLEASE TOKENS SPDU	ACTIVITY DISCARD SPDU ACTIVITY DISCARD ACK SPDU
GIVE TOKENS SPDU [a)] PLEASE TOKENS SPDU	ACTIVITY INTERRUPT SPDU ACTIVITY INTERRUPT ACK SPDU
GIVE TOKENS SPDU PLEASE TOKENS SPDU	ACTIVITY END SPDU ACTIVITY END ACK SPDU
GIVE TOKENS SPDU [a)] PLEASE TOKENS SPDU	CAPABILITY DATA SPDU CAPABILITY DATA ACK SPDU
PLEASE TOKENS SPDU PLEASE TOKENS SPDU	EXCEPTION REPORT SPDU EXCEPTION DATA SPDU

[a)] Indicates that the Token Item parameter is not present in the GIVE TOKENS SPDU. In all other cases, the Token Item parameter may or may not be present.

In all cases, the Token Item parameter may only be present in the first SPDU if the second SPDU contains either a complete SSDU, or the last segmented SSDU.

Basic concatenation of a PLEASE TOKENS SPDU or a GIVE TOKENS SPDU with a second SPDU is only permitted when the user data parameter is not present in the PLEASE TOKENS SPDU or the GIVE TOKENS SPDU.

TABLE 8/X.225

Valid extended concatenation of SPDUs

First SPDU	Second SPDU	Third SPDU	Fourth SPDU	Status
GIVE TOKENS SPDU	MINOR SYNC ACK SPDU			
GIVE TOKENS SPDU	MAJOR SYNC ACK SPDU			
GIVE TOKENS SPDU	ACTIVITY END ACK SPDU			
GIVE TOKENS SPDU	ACTIVITY START SPDU	MINOR SYNC POINT SPDU		
GIVE TOKENS SPDU	ACTIVITY RESUME SPDU	MINOR SYNC POINT SPDU		
GIVE TOKENS SPDU [a)]	ACTIVITY START SPDU	ACTIVITY END SPDU		
GIVE TOKENS SPDU [a)]	ACTIVITY RESUME SPDU	ACTIVITY END SPDU		
GIVE TOKENS SPDU [a)]	ACTIVITY START SPDU	MAJOR SYNC POINT SPDU		
GIVE TOKENS SPDU [a)]	ACTIVITY RESUME SPDU	MAJOR SYNC POINT SPDU		
GIVE TOKENS SPDU	MINOR SYNC POINT SPDU	DATA TRANSFER SPDU		CL
GIVE TOKENS SPDU	MINOR SYNC ACK SPDU	DATA TRANSFER SPDU		CL
GIVE TOKENS SPDU	MAJOR SYNC POINT SPDU	DATA TRANSFER SPDU		CL
GIVE TOKENS SPDU	MAJOR SYNC ACK SPDU	DATA TRANSFER SPDU		CL
GIVE TOKENS SPDU	ACTIVITY START SPDU	DATA TRANSFER SPDU		CF
GIVE TOKENS SPDU	ACTIVITY RESUME SPDU	DATA TRANSFER SPDU		CF
GIVE TOKENS SPDU	ACTIVITY END SPDU	DATA TRANSFER SPDU		CL
GIVE TOKENS SPDU	ACTIVITY END ACK SPDU	DATA TRANSFER SPDU		CL

TABLE 8/X.225 *(cont.)*

First SPDU	Second SPDU	Third SPDU	Fourth SPDU	Status
GIVE TOKENS SPDU	ACTIVITY START SPDU	MINOR SYNC POINT SPDU	DATA TRANSFER SPDU	C
GIVE TOKENS SPDU	ACTIVITY RESUME SPDU	MINOR SYNC POINT SPDU	DATA TRANSFER SPDU	C
GIVE TOKENS SPDU [a)]	ACTIVITY START SPDU	ACTIVITY END SPDU	DATA TRANSFER SPDU	C
GIVE TOKENS SPDU [a)]	ACTIVITY RESUME SPDU	ACTIVITY END SPDU	DATA TRANSFER SPDU	C
GIVE TOKENS SPDU [a)]	ACTIVITY START SPDU	MAJOR SYNC POINT SPDU	DATA TRANSFER SPDU	C
GIVE TOKENS SPDU [a)]	ACTIVITY RESUME SPDU	MAJOR SYNC POINT SPDU	DATA TRANSFER SPDU	C

[a)] Indicates that the Token Item parameter is not present in the GIVE TOKENS SPDU.

Status:

CL: The DATA TRANSFER SPDU contains a complete SSDU or the last segment of an SSDU.

CF: The DATA TRANSFER SPDU contains a complete SSDU or the first segment of an SSDU. In the latter case, the Token Item parameter is not present in the GIVE TOKENS SPDU.

C: The DATA TRANSFER SPDU contains a complete SSDU.

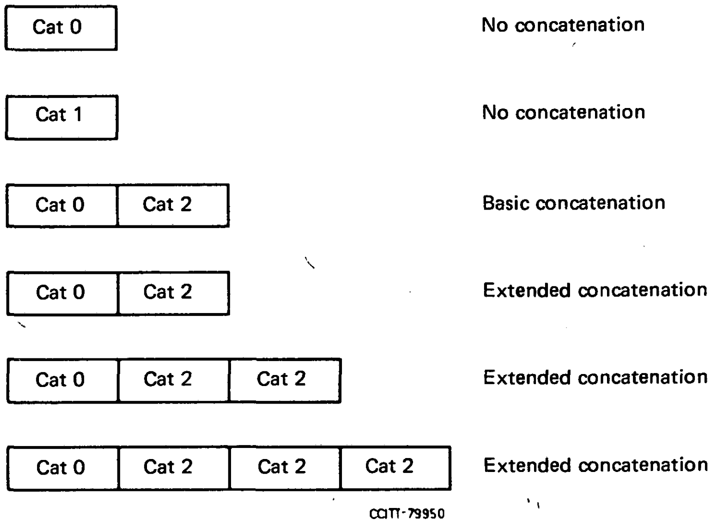


FIGURE 3/X.225

Illustration of TSDU structures

6.4 *Use of transport expedited data*

6.4.1 *Purpose*

To convey SPDUs on a separate transport flow.

6.4.2 *Transport service primitives*

The procedure uses the following transport service primitives:

T-EXPEDITED-DATA request
T-EXPEDITED-DATA indication

6.4.3 *SPDUs used*

The following SPDUs are sent on the transport expedited flow when it is available:

ABORT SPDU (see § 7.9);
ABORT ACCEPT SPDU (see § 7.10);
EXPEDITED DATA SPDU (see § 7.12);
PREPARE SPDU (see § 7.26).

6.4.4 *Description*

The SPDUs listed in § 6.4.3 are sent on the transport expedited flow if it is selected, and may be used to bypass any flow control restrictions or congestion on the transport normal flow. SPDUs sent on the transport expedited flow may be delivered to the accepting SS-user earlier than SSDUs submitted previously by the sending SS-user and sent on the transport normal flow, but no later than subsequently submitted SSDUs.

When the transport expedited flow is not available:

a) EXPEDITED DATA SPDUs are not sent;

b) ABORT and ABORT ACCEPT SPDUs are sent on the transport normal flow;

c) PREPARE SPDUs are not sent.

6.5 *Flow control*

There is no peer flow control in the session layer. To prevent the SS-users from being overloaded with data, the receiving SPM may apply back pressure across the transport connection, using the transport flow control. The decision on when or how back pressure is applied is a local matter.

6.6 *Transport disconnection*

6.6.1 *Purpose*

To release a transport connection.

6.6.2 *Transport service primitives*

The procedure uses the following transport service primitives:

T-DISCONNECT request
T-DISCONNECT indication

6.6.3 *SPDUs used*

No SPDUs are used.

6.6.4 *Description*

After the session connection has been released or aborted and the transport connection is not to be reused, the transport connection is disconnected.

When a T-DISCONNECT indication is received, as a result of an error detected by the transport service provider, the SPM issues an S-P-ABORT indication to the local SS-user.

When issuing a T-DISCONNECT request, the SPM may optionally use the T-DISCONNECT user data field to indicate the reason for the transport disconnection to the remote SPM. The reason code consists of one octet with the following values:

a) 0 – session protocol error for which an ABORT SPDU could not be sent;

b) 1 – normal transport disconnection when the transport connection is not to be reused;

c) 2 – normal transport disconnection when the transport connection was to be reused, but reuse is not possible for local reasons.

The use of the Disconnect Reason parameter in T-DISCONNECT indication is a local matter.

7 Elements of procedure related to SPDUs

This section defines valid sequences of operation of the protocol.

A more precise definition of procedures is contained in Annex A which incoporates all the checks to determine the validity of a particular event at a particular point in time. In case of arbitration or dispute, Annex A takes precedence over this section.

The elements of procedure specified in §§ 7.4-7.8, 7.14-7.18, 7.20-7.23 and 7.28-7.36 do not consider the case where an SSDU is segmented. (The circumstances under which an SSDU may be segmented are specified in § 6.3.5). Additional elements of procedure for segmented SSDUs are specified in § 7.37.

7.1 *CONNECT SPDU*

The CONNECT SPDU is transmitted by the initiator of the transport connection on a previously assigned transport connection in order to initiate a session connection.

7.1.1 *Content of CONNECT SPDU*

The CONNECT SPDU contains:

a) Connection Identifier parameter group, which is supplied by the calling SS-user, to enable the SS-users to identify this specific session connection. This parameter group has no effect on the SPM. It contains:

   1) Calling SS-user Reference parameter;

   2) Common Reference parameter;

   3) Additional Reference Information parameter.

b) Connect/Accept Item parameter group containing:

   1) Protocol Options parameter which enables the initiator to indicate its ability to receive extended concatenated SPDUs;

   2) TSDU maximum size parameter which, if present and not zero, indicates that the initiator's proposed values for the maximum TSDU sizes for each direction of transfer (see §§ 5.7.5 and 6.3.5). If this parameter is not present or is zero, the TSDU size is not limited.

   3) Version Number parameter to identify all versions of this protocol which are supported and are suitable for this session connection.

      Note – Protocol version 1 is not suitable if there are more than 512 octets of SS-user data in this SPDU.

4) Initial Serial Number parameter which is proposed by the calling SS-user in the case where the activity management functional unit is not proposed and any of the minor synchronize, major synchronize or resynchronize functional units are proposed. As an SS-user option, an Initial Serial Number parameter may be proposed even if the activity management functional unit is proposed provided that any of the minor synchronize, major synchronize or resynchronize functional units are also proposed;

5) Token Setting Item parameter supplied by the calling SS-user, which proposes the initial token positions for each token available on this connection, as derived from the functional units proposed in the Session User Requirements parameter (see Table 4/X.225). The initial token positions can be specified to be on the initiator's side or on the acceptor's side or the initiator can specify that the decision is to be made by the called SS-user.

c) Session User Requirements parameter containing a list of the functional units proposed by the calling SS-user. At least one of the half-duplex and the duplex functional untis shall be proposed. The SPM is required to provide the associated protocol functions.

d) Calling Session Selector and Called Session Selector parameters corresponding to the calling SS-user and the called SS-user may be present and are derived from session addresses provided by the calling SS-user.

e) Either a User Data parameter which allows a limited (512 or less octets) amount of transparent user data to be passed from the calling SS-user to the called SS-user or

an Extended User Data parameter which allows between 513 and 10 240 octets of transparent user data to be passed from the calling SS-user to the called SS-user. This parameter shall not be present if protocol version 1 is proposed.

Only one of these two parameters may be used on the CONNECT SPDU.

f) Data Overflow parameter which shall be present if and only if there is more than 10 240 octets of SS-user data and which indicates to the responder that there is more SS-user data to follow. The first 10240 octets of SS-user data are sent in the Extended User Data parameter. This parameter shall not be present if protocol version 1 is proposed.

7.1.2 *Sending the CONNECT SPDU*

An S-CONNECT request results in the assignment of a transport connection. When the transport connection is established, a CONNECT SPDU is sent on the transport normal flow. If the Data Overflow parameter was not present in the CONNECT SPDU, the SPM waits until it receives an ACCEPT SPDU or a REFUSE SPDU. If the Data Overflow parameter was present in the CONNECT SPDU, the SPM waits until it receives an OVERFLOW ACCEPT or a REFUSE SPDU.

7.1.3 *Receiving the CONNECT SPDU*

A valid incoming CONNECT SPDU which is acceptable to the receiving SPM and does not contain the Data Overflow parameter results in an S-CONNECT indication to an SS-user, according to the Called Session Selector parameter of the CONNECT SPDU. The SPM then waits for an S-CONNECT response from the called SS-user.

A valid incoming CONNECT SPDU which is acceptable to the receiving SPM, contains the Data Overflow parameter and provided that protocol version 2 is to be selected, results in the SPM sending an OVERFLOW ACCEPT SPDU; the SPM then waits until it receives a CONNECT DATA OVERFLOW SPDU. Otherwise the SPM sends a REFUSE SPDU (see § 7.5).

7.2 *OVERFLOW ACCEPT SPDU*

The OVERFLOW ACCEPT SPDU is used by the SPM to request the remainder of the S-CONNECT request SS-user data.

7.2.1 *Content of the OVERFLOW ACCEPT SPDU*

The OVERFLOW ACCEPT SPDU contains:

a) TSDU Maximum Size parameter which, if present and not zero indicates that segmenting has been proposed by the responder (see § 6.3.5). The responder proposes alternative values for the maximum TSDU sizes for each direction of transfer (see § 5.7.5). These values may be larger than, smaller than or equal to the values supplied by the initiator in the CONNECT SPDU. The smaller value for each direction of transfer is used for the maximum TSDU size for that direction of transfer;

b) Version Number parameter indicating that at least protcol version 2 is supported.

7.2.2 *Sending the OVERFLOW ACCEPT SPDU*

A valid incoming CONNECT SPDU which contains the Data Overflow parameter results in the SPM sending an OVERFLOW ACCEPT SPDU. The SPM then waits until it receives a CONNECT DATA OVERFLOW SPDU.

7.2.3 *Receiving the OVERFLOW ACCEPT SPDU*

A valid incoming OVERFLOW ACCEPT SPDU results in the SPM sending one or more CONNECT DATA OVERFLOW SPDUs. When the last CONNECT DATA OVERFLOW SPDU has been sent, the SPM waits until it receives an ACCEPT SPDU or a REFUSE SPDU.

7.3 *CONNECT DATA OVERFLOW SPDU*

The CONNECT DATA OVERFLOW SPDU is used by the initiator to send subsequent segments of the User Data associated with an S-CONNECT request.

7.3.1 *Content of the CONNECT DATA OVERFLOW SPDU*

The CONNECT DATA OVERFLOW SPDU contains:

a) an Enclosure Item parameter to indicate whether the SPDU is the middle or the end of the SSDU;

b) a User data parameter which allows a maximum of 65528 octets of transparent user data to be transferred.

7.3.2 *Sending the CONNECT DATA OVERFLOW SPDU*

A valid incoming OVERFLOW ACCEPT SPDU results in the SPM sending one or more CONNECT DATA OVERFLOW SPDUs. These SPDUs will be sent as an ordered sequence with the appropriate value for the Enclosure Item parameter until the complete SSDU has been transferred.

7.3.3 *Receiving the CONNECT DATA OVERFLOW SPDU*

A valid incoming CONNECT DATA OVERFLOW SPDU with Enclosure Item parameter indicating "end of SSDU" results in an S-CONNECT indication to pass the entire SSDU to the SS-user, according to the Called Session selector parameter of the CONNECT SPDU. The SPM then waits for an S-CONNECT response from the called SS-user.

If the Enclosure Item parameter in a valid incoming CONNECT DATA OVERFLOW SPDU indicates "not end of SSDU", the SPM waits for a subsequent valid CONNECT DATA OVERFLOW SPDU.

7.4 *ACCEPT SPDU*

An SPM receiving a CONNECT SPDU with the Data Overflow parameter absent may accept a proposal to establish a session connection by transferring an ACCEPT SPDU (after receiving an S-CONNECT response primitive) to the initiator, on the same transport connection.

An SPM which has previously issued an OVERFLOW ACCEPT SPDU in response to a CONNECT SPDU with the Data Overflow parameter present and which subsequently receives the sequence of CONNECT DATA OVERFLOW SPDUs which complete the segmented SSDU may accept a proposal to establish a session connection by trasferring an ACCEPT SPDU (after receiving an S-CONNECT response primitive) to the initiator, on the same transport connection.

7.4.1 *Content of ACCEPT SPDU*

The ACCEPT SPDU contains:

a) Connection Identifier parameter group, which is supplied by the called SS-user, to enable the SS-users to identify this specific session connection. This parameter group has no effect on the SPM. It contains:

   1) Called SS-user Reference parameter;

   2) Common Reference parameter;

   3) Additional Reference Information parameter.

b) Connect/Accept Item parameter group containing:

   1) Protocol Options parameter which allows the responder to indicate its ability to receive extended concatenated SPDUs;

   2) TSDU maximum size parameter which, if present and not zero, indicates the responder's proposed values for the maximum TSDU sizes for each direction of transfer (see §§ 5.7.6 and 6.3.5). These values may be larger than, smaller than or equal to the values supplied by the initiator in the CONNECT SPDU. The smaller value is used for the maximum TSDU size for each direction of transfer. If an OVERFLOW ACCEPT SPDU has previously been sent on this session connection then:

      i) if the TSDU Maximum Size parameter was present in the OVERFLOW ACCEPT SPDU, then it shall also be present in the ACCEPT SPDU, with the same values as were given in the OVERFLOW ACCEPT SPDU.
      ii) if the TSDU Maximum Size parameter was not present in the OVERFLOW ACCEPT SPDU, then it shall not be present in the ACCEPT SPDU;

   3) Version Number parameter to identify all versions of this protocol which are supported and are suitable for this session connection. If an OVERFLOW ACCEPT SPDU has been previously sent on this session connection then the Version Number parameter shall be present in the ACCEPT SPDU, with the same value as was given in the OVERFLOW ACCEPT SPDU. The highest version number indicated by both initiator and responder is used;

      Note – Protocol version 1 is not suitable if there are more than 512 octets of User Data in this SPDU.

   4) Initial Serial Number parameter which is present if the activity management functional unit is not selected and any of the minor synchronize, major synchronize or resynchronize functional units are selected regardless of whether or not the activity management functional unit is proposed. The called SS-user proposes the value, which is the value of the first serial number to be used;

   5) Token Setting Item parameter supplied by the called SS-user, which indicates the initial token positions for each token available on this session connection, as derived from the selected functional units. A token is only available if any functional unit which requires that token has been selected for use on this session connection (see Table 4/X.225), regardless of the settings of the Token Setting Item parameter in the CONNECT SPDU (see § 7.1.1 b) 5)). If a token-controlled functional unit has been selected, then in the case where the calling SS-user has indicated that the initial assignment of the related token is the called SS-user's choice, this parameter contains a value chosen by the called SS-user. Otherwise, the values indicated by the calling SS-user in the CONNECT SPDU are selected and must be returned.

c) Token Item parameter which allows the called SS-user to request tokens which have been assigned to the calling SS-user in the CONNECT SPDU.

d) Session User Requirements parameter which contains a list indicating the functional units proposed by the called SS-user and can be supported by the responder. The functional units selected for use on this session connection are the intersection of this set and the set proposed in the CONNECT SPDU (i.e. only those functional units indicated in both the CONNECT SPDU and the ACCEPT SPDU are selected). If both the half-duplex functional unit and the duplex functional unit were indicated in the CONNECT SPDU, then the ACCEPT SPDU must propose which one is to be available. If only one of these functional units was indicated in the CONNECT SPDU, then the ACCEPT SPDU must indicate that the same functional unit is to be used (or the connection attempt must be rejected).

e) Calling Session Selector parameter corresponding to the calling SS-user may be present, in which case it will have the same value as in the CONNECT SPDU. Responding Session Selector parameter corresponding to the responding Session Address provided by the responding SS-user.

f) User Data parameter which allows a limited amount of transparent user data to be passed from the called SS-user to the calling SS-user.

7.4.2 *Sending the ACCEPT SPDU*

An S-CONNECT (accept) response results in an ACCEPT SPDU. This SPDU is sent on the transport normal flow. After this successful connection, the SPM enters the data transfer phase and can receive any service request or SPDU that is allowed by the selected functional units and current token positions. If any of the minor synchronize, major synchronize or resynchronize functional units are selected but the activity management functional unit is not selected, the SPM sets V(A) and V(M) to the Initial Serial Number proposed by the called SS-user, which is the serial number to be used for the first synchronization point. V(R) is set to zero. Vsc is set false. If the activity management functional unit has been selected, Vact is set false.

7.4.3 *Receiving the ACCEPT SPDU*

A valid incoming ACCEPT SPDU results in an S-CONNECT (accept) confirm. After this successful connection, the SPM enters the data transfer phase and can receive any service request or SPDU that is allowed by the available functional units and current token positions. If any of the minor synchronize, major synchronize or resynchronize functional units are selected but the activity management functional unit is not selected, the SPM sets V(A) and V(M) to the Initial Serial Number contained in the ACCEPT SPDU, which is the serial number to be used for the first synchronization point. V(R) is set to zero. Vsc is set false. If the activity management functional unit has been selected, Vact is set false.

If the called SS-user has requested any tokens in the Token Item parameter of the ACCEPT SPDU (see § 7.4.1 c)), an S-PLEASE-TOKEN indication is also generated.

7.5 *REFUSE SPDU*

A REFUSE SPDU is used by the responder to reject an attempt to establish a session connection.

7.5.1 *Content of REFUSE SPDU*

The REFUSE SPDU contains:

a) Connection Identifier parameter group, which is supplied by the called SS-user, to enable the SS-users to identify this specific session connection. This parameter group has no effect on the SPM. It contains:

   1) Called SS-user Reference parameter;
   2) Common Reference parameter;
   3) Additional Reference Information parameter.

b) Transport Disconnect parameter which indicates whether or not the transport connection is to be kept.

c) Session User Requirements parameter which contains a list of the functional units supported by the sending SPM, and required by the called SS-user.

d) Version Number parameter to identify which versions of this protocol have been implemented by the sending SPM.

e) Reason Code parameter giving the reason for refusal of the attempt to establish a session connection, together with a limited amount of transparent user data.

7.5.2 *Sending the REFUSE SPDU*

An S-CONNECT (reject) response results in a REFUSE SPDU. This SPDU is sent on the transport normal flow. No session connection is established. If the Transport Disconnect parameter indicates that the transport connection can be reused, the SPM waits for a CONNECT SPDU. Otherwise, the SPM starts the timer, TIM, and waits for a T-DISCONNECT indication. If the timer expires before receipt of a T-DISCONNECT indication, the SPM requests transport disconnection with a T-DISCONNECT request. The timer is cancelled on receipt of a T-DISCONNECT indication.

Note – The value of TIM is a local implementation dependent matter, related to quality of service.

7.5.3 *Receiving the REFUSE SPDU*

A valid incoming REFUSE SPDU results in an S-CONNECT (reject) confirm. No session connection is established. If the Transport Disconnect parameter indicates that retention of the transport connection has been requested by the called SPM, and this is acceptable to the calling SPM, the SPM waits for an S-CONNECT request. Otherwise, the SPM releases the transport connection, by making a T-DISCONNECT request.

7.6 *FINISH SPDU*

Orderly release is initiated by transfer of a FINISH SPDU, which may be transferred during the data transfer phase. It requests as a response either:

a) a DISCONNECT SPDU to complete the release of the session connection, or

b) a NOT FINISHED SPDU to refuse the release of the session connection if the release token is available.

The FINISH SPDU is transferred in sequence with any normal data being transferred. The right to issue a FINISH SPDU is restricted to the owner of all available tokens.

7.6.1 *Content of FINISH SPDU*

The FINISH SPDU contains:

a) Transport Disconnect parameter which indicates whether or not the transport connection is to be kept, subject to the restrictions specified in § 6.2.4;

b) User Data parameter which allows a limited amount of transparent user data to be transferred.

7.6.2 *Sending the FINISH SPDU*

An S-RELEASE request results in a FINISH SPDU. This SPDU is sent on the transport normal flow. After transferring a FINISH SPDU, the SPM may not send any further SPDUs (except ABORT SPDU or, in the case of collision of FINISH SPDUs, a DISCONNECT SPDU) unless a NOT FINISHED SPDU or a RESYNCHRONIZE SPDU is received, after which the data transfer phase may be resumed. Receipt of a DISCONNECT SPDU signals completion of orderly session release.

7.6.3 *Receiving the FINISH SPDU*

A valid incoming FINISH SPDU results in an S-RELEASE indication. The user data is passed to the SS-user. The SPM waits for an S-RELEASE response.

7.7 *DISCONNECT SPDU*

After receipt of a FINISH SPDU, a DISCONNECT SPDU may be transferred. Receipt of a DISCONNECT SPDU after transferring a FINISH SPDU, signals the orderly release of the session connection. The DISCONNECT SPDU is transferred in sequence with any normal data being transferred.

7.7.1 *Content of DISCONNECT SPDU*

The DISCONNECT SPDU contains a User Data parameter which allows a limited amount of transparent user data to be transferred.

7.7.2 *Sending the DISCONNECT SPDU*

An S-RELEASE (accept) response results in a DISCONNECT SPDU. This SPDU is sent on the transport normal flow. The session connection ceases to exist.

If the FINISH SPDU indicated that the transport connection is to be kept for reuse, and this is acceptable, the SPM waits for a CONNECT SPDU. Otherwise, the SPM starts the timer, TIM, and waits for a T-DISCONNECT indication. If the timer expires before receipt of a T-DISCONNECT indication, the SPM requests transport disconnection with a T-DISCONNECT request. The timer is cancelled on receipt of a T-DISCONNECT indication.

Note – The value of TIM is a local implementation dependent matter, related to quality of service.

7.7.3 *Receiving the DISCONNECT SPDU*

A valid incoming DISCONNECT SPDU results in an S-RELEASE (accept) confirm. The session connection ceases to exist.

If the transport connection is to be kept for reuse (see § 6.2.4), the SPM waits for a suitable S-CONNECT request. Otherwise, a T-DISCONNECT request is issued.

Note 1 – In the case of collision of FINISH SPDU and ABORT SPDU (see § 7.9), the ABORT SPDU takes preference and thus the indication in the FINISH SPDU to keep or release the transport connection is ignored.

Note 2 – In the case of collision of FINISH SPDUs (data token and release token not available) the transport connection cannot be reused. The SPM receiving the DISCONNECT SPDU issues a T-DISCONNECT request.

7.8 *NOT FINISHED SPDU*

After receipt of a FINISH SPDU, a NOT FINISHED SPDU may be transferred subject to the token restrictions specified in Table 5/X.225. No confirmation is sought.

7.8.1 *Content of NOT FINISHED SPDU*

The NOT FINISHED SPDU contains a User Data parameter which allows a limited amount of transparent user data to be transferred.

7.8.2 *Sending the NOT FINISHED SPDU*

An S-RELEASE (reject) response results in a NOT FINISHED SPDU. This SPDU is sent on the transport normal flow. The SPM remains in the data transfer phase and can receive any service request or SPDU that is allowed by the available functional units and current token positions.

7.8.3 *Receiving the NOT FINISHED SPDU*

A valid incoming NOT FINISHED SPDU results in an S-RELEASE (reject) confirm. The SPM remains in the data transfer phase and can receive any service request or SPDU that is allowed by the available functional units and current token positions.

7.9 *ABORT SPDU*

The ABORT SPDU is used to reject a session connection establishment attempt, or to cause abnormal release of a session connection at any time. This SPDU is also used by an SPM to release the session connection when a protocol error is detected. The ABORT SPDU may or may not request that the transport connection be released by the receiving SPM. Use of the ABORT SPDU may result in loss of data.

7.9.1 *Content of ABORT SPDU*

7.9.1.1 If there is no SSDU or segmenting of the SSDU is not required (see § 6.3.5), the ABORT SPDU contains:

a) a Transport Disconnect parameter which indicates whether or not the transport connection is to be kept;

b) a Reflect Parameter Values parameter which, if present, allows implementation defined information to be transferred;

c) a User Data parameter which, if present, allows a limited amount of transparent user data to be transferred.

7.9.1.2 If the SSDU is to be segmented, the first ABORT SPDU contains:

a) a Transport Disconnect parameter which indicates whether or not the transport connection is to be kept;

b) an Enclosure Item parameter which indicates that this SPDU is the beginning of the SSDU and not the end of the SSDU;

c) a User Data parameter which allows a limited amount of transparent user data to be transferred.

The second and any subsequent ABORT SPDUs in the sequence of ABORT SPDUs transmitting the SSDU contain:

d) an Enclosure Item parameter to indicate whether the SPDU is the middle or end of the SSDU;

e) a User Data parameter which allows a limited amount of transparent user data to be transferred.

7.9.2 *Sending the ABORT SPDU*

An S-U-ABORT request or the detection of a protocol error in any state of the SPM results in either a single ABORT SPDU or, if the SSDU provided in the S-U-ABORT request is to be segmented (see § 6.3.5), a sequence of ABORT SPDUs, which shall not be interrupted.

If the SS-user data does not exceed 9 octets, the ABORT SPDU is sent on the transport expedited flow, if it is available to this session connection. If the transport expedited flow is not available to this session connection this SPDU is sent on the transport normal flow.

If the SS-user data exceeds 9 octets, the SPDU or sequence of SPDUs are sent on the transport normal flow. If the transport expedited flow is available to this session connection, a PREPARE (ABORT) SPDU is sent on the transport expedited flow simultaneously or earlier than the first or only ABORT SPDU. The SPM starts the timer, TIM, and waits for an ABORT ACCEPT SPDU or a T-DISCONNECT indication. Any other SPDUs are discarded. If the timer expires before receipt of an ABORT ACCEPT SPDU or a T-DISCONNECT indication, the SPM shall request transport disconnection with a T-DISCONNECT request. On receipt of a T-DISCONNECT indication, the timer is cancelled.

Note – The value of TIM is a local implementation dependent matter, related to quality of service.

7.9.3 *Receiving the ABORT SPDU*

A valid incoming ABORT SPDU, without an Enclosure Item parameter, or an Enclosure Item parameter indicating "end of SSDU" results in an S-U-ABORT indication or an S-P-ABORT indication, depending on whether the abort is user generated or provider generated. The session connection ceases to exist. If the Transport Disconnect parameter in the received ABORT SPDU indicates that the transport connection is to be kept for reuse and this is acceptable to the receiving SPM, an ABORT ACCEPT SPDU is sent. If the Transport Disconnect parameter in the received ABORT SPDU indicates that the transport connection is not to be kept for reuse or reuse of the transport connection is not acceptable to the receiving SPM, the receiving SPM either:

a) releases the transport connection, or

b) sends an ABORT ACCEPT SPDU (see § 7.10).

Receiving an ABORT SPDU sent in response to a CONNECT SPDU results in:

a) a T-DISCONNECT request, unless retention of the transport connection has been requested in the ABORT SPDU, in which case the ABORT SPDU is acknowledged with an ABORT ACCEPT SPDU (see § 7.10); and

b) an S-P-ABORT indication or an S-U-ABORT indication to the SS-user.

7.10 *ABORT ACCEPT SPDU*

The ABORT ACCEPT SPDU is used to return a confirmation to the ABORT SPDU.

7.10.1 *Content of ABORT ACCEPT SPDU*

The ABORT ACCEPT SPDU contains no parameters.

7.10.2 *Sending the ABORT ACCEPT SPDU*

A valid incoming ABORT SPDU results in sending an ABORT ACCEPT SPDU, when the transport connection can be reused, i.e. when:

a) the transport expedited service is not available to this session connection, and

b) retention of the transport connection has been requested in the ABORT SPDU and it is acceptable to reuse the transport connection.

The SPM, as a local implementation decision, may send an ABORT ACCEPT SPDU in response to an ABORT SPDU, even if the transport connection is not to be kept.

This SPDU is sent on the transport expedited flow, if it is available to this session connection. Otherwise, this SPDU is sent on the transport normal flow. The session connection ceases to exist.

7.10.3 *Receiving the ABORT ACCEPT SPDU*

A valid incoming ABORT ACCEPT SPDU results in resetting the timer, TIM, and:

a) releasing the transport connection, if release of the transport connection was requested in the previously sent ABORT SPDU;

b) if retention of the transport connection was requested, the transport connection is now available for reuse by a new session connection, if this SPM was the initiator of the transport connection (see § 6.1).

The session connection ceases to exist.

7.11 *DATA TRANSFER SPDU*

Normal data is transferred by use of the DATA TRANSFER SPDU. If the extended concatenation option was selected during connection establishment, certain concatenations of the DATA TRANSFER SPDU with other SPDUs is allowed (see § 6.3.7).

The right to issue a DATA TRANSFER SPDU is subject to the token restrictions specified in Table 5/X.225.

7.11.1 *Content of DATA TRANSFER SPDU*

The DATA TRANSFER SPDU contains:

a) Enclosure Item parameter to indicate the beginning and end of SSDU when segmenting has been selected. When segmenting has been selected, the Enclosure Item parameter is always present and indicates whether the SPDU is the beginning, middle or end of the SSDU. When segmenting has not been selected, the Enclosure Item parameter is not present;

b) User Information Field to transfer transparent user data whose maximum size is unlimited when segmenting has not been selected and whose maximum size is limited by the maximum TSDU size when segmenting has been selected.

7.11.2 *Sending the DATA TRANSFER SPDU*

An S-DATA request results in a DATA TRANSFER SPDU unless segmenting has been selected, in which case an ordered sequence of DATA TRANSFER SPDUs will be sent with the appropriate value for the Enclosure Item parameter until the complete SSDU has been transferred.

The concatenation of any segment of an SSDU with any other SPDU will not result in a TSDU larger than the selected maximum TSDU size for that direction of transfer. However, there is no requirement that the resulting TSDU should be of the maximum size for that direction of transfer. All DATA TRANSFER SPDUs, except the last DATA TRANSFER SPDU in a sequence greater than one, must have user information. DATA TRANSFER SPDUs are sent on the transport normal flow.

Sending a segmented SSDU shall be interrupted when the SPM which is sending the segmented SSDU sends or receives one of:

RESYNCHRONIZE SPDU

EXCEPTION REPORT SPDU

EXCEPTION DATA SPDU

ACTIVITY INTERRUPT SPDU

ACTIVITY DISCARD SPDU

ABORT SPDU

PREPARE (RESYNCHRONIZE) SPDU

PREPARE (ABORT) SPDU

or receives a T-DISCONNECT indication. This will have a destructive effect on the entire SSDU. The SPM is not required to send the remainder of the ordered sequence of SPDUs which comprise the segmented SSDU (but may do so if it wishes).

7.11.3 *Receiving the DATA TRANSFER SPDU*

A valid incoming DATA TRANSFER SPDU results in an S-DATA indication unless segmenting has been selected. In this case, a valid incoming DATA TRANSFER SPDU, which indicates end of SSDU, results in an S-DATA indication to pass the entire SSDU to the SS-user.

Where segmenting has been selected and an incomplete segmented SSDU is outstanding, the receipt of:

RESYNCHRONIZE SPDU
EXCEPTION REPORT SPDU
EXCEPTION DATA SPDU
ACTIVITY INTERRUPT SPDU
ACTIVITY DISCARD SPDU
ABORT SPDU
PREPARE (RESYNCHRONIZE) SPDU

has a destructive effect on the entire SSDU (i.e. the SPDUs which have already been received are discarded, the remaining SPDUs will not be received).

Receiving a segmented SSDU shall be interrupted when the SPM which is receiving the segmented SSDU sends or receives one of:

RESYNCHRONIZE SPDU
EXCEPTION REPORT SPDU
EXCEPTION DATA SPDU
ACTIVITY INTERRUPT SPDU
ACTIVITY DISCARD SPDU
ABORT SPDU
PREPARE (RESYNCHRONIZE) SPDU
PREPARE (ABORT) SPDU

or receives a T-DISCONNECT indication. This will have a destructive effect on the entire SSDU (i.e. the SPDUs comprising part of the segmented SSDU which have already been received are discarded, and any SPDUs comprising part of the segmented SSDU which are received subsequently are discarded).

The receipt of any other SPDUs is a protocol error.

7.12 *EXPEDITED SPDU*

The EXPEDITED SPDU is used to transfer expedited SSDUs.

The right to send expedited data is not associated with any tokens. When this functional unit is selected, both SS-users may send expedited data. An EXPEDITED SSDU may be delivered to the receiving SS-user prior to other SSDUs previously transferred on the transport normal flow; it may not be delivered to the receiving SS-user later than any SSDUs transferred after it.

Expedited SSDUs are delivered to the receiving SS-user in the same sequence in which they were issued by the sending SS-user.

7.12.1 *Content of EXPEDITED SPDU*

The EXPEDITED SPDU contains a User Information Field which allows a limited amount of transparent user data to be transferred.

7.12.2 *Sending the EXPEDITED SPDU*

An S-EXPEDITED-DATA request results in an EXPEDITED SPDU being sent. This SPDU is sent on the transport expedited flow.

7.12.3 *Receiving the EXPEDITED SPDU*

A valid incoming EXPEDITED SPDU results in an S-EXPEDITED-DATA indication.

7.13 *TYPED DATA SPDU*

The TYPED DATA SPDU enables the SS-users to transmit transparent user data, irrespective of the availability or assignment of the data token. In all other respects, the same constraints apply as for normal data (see § 7.11). The same rules for segmenting also apply.

7.13.1 *Content of TYPED DATA SPDU*

The TYPED DATA SPDU contains:

a) Enclosure Item parameter to indicate the beginning and end of SSDU when segmenting has been selected. When segmenting has been selected, the Enclosure Item parameter is always present and indicates whether the SPDU is the beginning, middle or end of the SSDU. When segmenting has not been selected, the Enclosure Item parameter is not present;

b) User Information Field to transfer transparent user data whose maximum size is unlimited when segmenting has not been selected and whose maximum size is limited by the maximum TSDU size when segmenting has been selected.

7.13.2 *Sending the TYPED DATA SPDU*

An S-TYPED-DATA request results in the transfer of a TYPED DATA SPDU unless segmenting has been selected, in which case an ordered sequence of TYPED DATA SPDUs will be sent with the appropriate value for the Enclosure Item parameter until the complete SSDU has been transferred. Each SPDU is mapped onto one TSDU and will not be larger than the selected maximum TSDU size for that direction of transfer. However, there is no requirement that the resulting TSDU should be of the maximum size for that direction of transfer. All TYPED DATA SPDUs, except the last TYPED DATA SPDU in a sequence greater than one, must have user information. TYPED DATA SPDUs are sent on the transport normal flow. When segmenting has been selected the rules governing the sending or receipt of SPDUs other than TYPED DATA SPDUs, while sending a segmented TYPED DATA SSDU are the same as for the DATA TRANSFER SPDU (see § 7.11.2).

7.13.3 *Receiving the TYPED DATA SPDU*

A valid incoming TYPED DATA SPDU results in an S-TYPED-DATA indication, unless segmenting has been selected. In this case, a valid incoming TYPED DATA SPDU which indicates end of SSDU results in an S-TYPED-DATA indication to pass the entire SSDU to the SS-user. The current state of the SPM is not changed.

When segmenting has been selected the rules governing the sending or receipt of SPDUs other than TYPED DATA SPDUs while receiving a segmented TYPED DATA SSDU are the same as for the DATA TRANSFER SPDU (see § 7.11.3).

7.14 *CAPABILITY DATA SPDU*

The CAPABILITY DATA SPDU is used to transfer a limited amount of transparent user data outside activities (i.e. when the activity management functional unit has been selected and Vact is false). The right to send this SPDU is restricted to the side having the right to start the next activity (i.e. the activity management functional unit has been selected and Vact is false and subject to the token restrictions specified in Table 5/X.225).

7.14.1 *Content of CAPABILITY DATA SPDU*

The CAPABILITY DATA SPDU contains a User Data parameter which allows a limited amount of transparent user data to be transferred.

7.14.2 *Sending the CAPABILITY DATA SPDU*

An S-CAPABILITY-DATA request results in a CAPABILITY DATA SPDU being sent. This SPDU is sent on the transport normal flow. The SS-user is not permitted to issue a further S-CAPABILITY-DATA request until this CAPABILITY DATA SPDU is acknowledged.

7.14.3 *Receiving the CAPABILITY DATA SPDU*

A valid incoming CAPABILITY DATA SPDU results in an S-CAPABILITY-DATA indication to the SS-user.

7.15 *CAPABILITY DATA ACK SPDU*

The CAPABILITY DATA ACK SPDU is used to complete the capability data exchange.

7.15.1 *Content of CAPABILITY DATA ACK SPDU*

The CAPABILITY DATA ACK SPDU contains a User Data parameter which allows a limited amount of transparent user data to be transferred.

7.15.2 *Sending the CAPABILITY DATA ACK SPDU*

The SS-user generates an S-CAPABILITY-DATA response which results in a CAPABILITY DATA ACK SPDU. This SPDU is sent on the transport normal flow.

7.15.3 *Receiving the CAPABILITY DATA ACK SPDU*

A valid incoming CAPABILITY DATA ACK SPDU results in an S-CAPABILITY-DATA confirm. This allows the SS-user to issue a further S-CAPABILITY-DATA request.

7.16 *GIVE TOKENS SPDU*

The GIVE TOKENS SPDU is used:

a) to introduce a concatenated sequence of SPDUs; and/or

b) to cause assignment of currently owned tokens to be changed.

If the GIVE TOKENS SPDU does not contain any parameter fields, it is used to indicate concatenation without assignment of tokens and, in this case, the sending and receiving procedures do not apply.

7.16.1 *Content of GIVE TOKENS SPDU*

The GIVE TOKENS SPDU contains:

a) a Token Item parameter which indicates which tokens are being transferred from the sending SS-user to the receiving SS-user.

b) a User Data parameter which allows a limited amount of transparent user data to be transferred. This parameter shall not be present if protocol version 1 is selected.

7.16.2 *Sending the GIVE TOKENS SPDU*

An S-TOKEN-GIVE request results in a GIVE TOKENS SPDU. This SPDU is sent on the transport normal flow.

7.16.3 *Receiving the GIVE TOKENS SPDU*

A valid incoming GIVE TOKENS SPDU results in an S-TOKEN-GIVE indication.

7.17 *PLEASE TOKENS SPDU*

The PLEASE TOKENS SPDU is used:

a) to introduce a concatenated sequence of SPDUs; and/or

b) to request that the token assignments be changed to permit the requestor to be authorized to perform a function associated with the requested tokens.

If the PLEASE TOKENS SPDU does not contain any parameter fields, it is used to indicate concatenation without requesting tokens and, in this case, the sending and receiving procedures do not apply.

7.17.1 *Content of PLEASE TOKENS SPDU*

The PLEASE TOKENS SPDU contains:

a) Token Item parameter which indicates which tokens are being requested by the sending SS-user;

b) User Data parameter which allows a limited amount of transparent user data to be transferred.

7.17.2 *Sending the PLEASE TOKENS SPDU*

An S-TOKEN-PLEASE request results in a PLEASE TOKENS SPDU. This SPDU is sent on the transport normal flow.

7.17.3 *Receiving the PLEASE TOKENS SPDU*

A valid incoming PLEASE TOKENS SPDU results in an S-TOKEN-PLEASE indication. Receiving a PLEASE TOKENS SPDU for tokens which are not currently assigned to the accepting SS-user is not a protocol error.

7.18 *GIVE TOKENS CONFIRM SPDU*

The GIVE TOKENS CONFIRM SPDU is used as a result of an S-CONTROL-GIVE request to cause assignment of all of the currently assigned tokens to be changed when Vact is false. Receipt of the GIVE TOKENS CONFIRM SPDU by the receiving SPM is acknowledged by the GIVE TOKENS ACK SPDU.

7.18.1 *Content of GIVE TOKENS CONFIRM SPDU*

The GIVE TOKENS CONFIRM SPDU contains a User Data parameter which allows a limited amount of transparent user data to be transferred. This parameter shall not be present if protocol version 1 is selected.

7.18.2 *Sending the GIVE TOKENS CONFIRM SPDU*

An S-CONTROL-GIVE request when Vact is false results in a GIVE TOKENS CONFIRM SPDU. The SPM then waits for a GIVE TOKENS ACK SPDU before permitting further SPDUs, associated with the available tokens, to be sent or received. SPDUs not associated with tokens (e.g. TYPED DATA SPDU) may be sent or received as normal. This SPDU is sent on the transport normal flow.

7.18.3 *Receiving the GIVE TOKENS CONFIRM SPDU*

A valid incoming GIVE TOKENS CONFIRM SPDU results in an S-CONTROL-GIVE indication, followed by a GIVE TOKENS ACK SPDU.

7.19 *GIVE TOKENS ACK SPDU*

The GIVE TOKENS ACK SPDU is used to acknowledge receipt of a GIVE TOKENS CONFIRM SPDU. The GIVE TOKENS ACK SPDU can only be sent when Vact is false.

7.19.1 *Content of GIVE TOKENS ACK SPDU*

The GIVE TOKENS ACK SPDU contains no parameters.

7.19.2 *Sending the GIVE TOKENS ACK SPDU*

A valid incoming GIVE TOKENS CONFIRM SPDU results in a GIVE TOKENS ACK SPDU (see also § 7.18.3). The SPM may now transmit SPDUs associated with the token controlled functional units. This SPDU is sent on the transport normal flow.

7.19.3 *Receiving the GIVE TOKENS ACK SPDU*

After receiving a valid incoming GIVE TOKENS ACK SPDU, the SPM is now prepared to receive any SPDUs associated with the token controlled functional units.

7.20 *MINOR SYNC POINT SPDU*

The MINOR SYNC POINT SPDU is used to define a minor synchronization point. A confirmation may be returned by the receiver but is not required by the SPM (see § 7.21). All acknowledgement rules are defined by the SS-users. In particular, whether confirmation is requested or not is transparent to the SPM. The right to issue a MINOR SYNC POINT SPDU is subject to the token restrictions specified in Table 5/X.225.

7.20.1 *Content of MINOR SYNC POINT SPDU*

The MINOR SYNC POINT SPDU contains:

a) Sync Type Item parameter which is used to indicate if an explicit confirmation is required (see § 7.21);

b) Serial Number parameter which indicates the serial number of this minor synchronization point, and is set by the SPM to the current value of V(M);

c) User Data parameter which allows a limited amount of transparent user data to be transferred.

7.20.2 *Sending the MINOR SYNC POINT SPDU*

An S-SYNC-MINOR request results in a MINOR SYNC POINT SPDU. This SPDU is sent on the transport normal flow. If Vsc is true, V(A) is set equal to V(M) and Vsc is set false. V(M) is incremented by one.

7.20.3 *Receiving the MINOR SYNC POINT SPDU*

A valid incoming MINOR SYNC POINT SPDU results in an S-SYNC-MINOR indication. If Vsc is false, V(A) is set equal to V(M) and Vsc is set true. V(M) is incremented by one.

7.21 *MINOR SYNC ACK SPDU*

The MINOR SYNC ACK SPDU is used to return a confirmation to minor synchronization points. The SPM sends a MINOR SYNC ACK SPDU for each S-SYNC-MINOR response.

7.21.1 *Content of MINOR SYNC ACK SPDU*

The MINOR SYNC ACK SPDU contains:

a) Serial Number parameter, provided by the SS-user which indicates the serial number of the minor synchronization point which is being confirmed;

b) User Data parameter which allows a limited amount of transparent user data to be transferred.

7.21.2 *Sending the MINOR SYNC ACK SPDU*

An S-SYNC-MINOR response (with Vsc true and serial number greater than or equal to V(A) and less than V(M)) results in sending a MINOR SYNC ACK SPDU. This SPDU is sent on the transport normal flow. The SPM sets V(A) equal to the serial number plus one.

7.21.3 *Receiving the MINOR SYNC ACK SPDU*

A valid incoming MINOR SYNC ACK SPDU (with Vsc false and received serial number greater than or equal to V(A) and less than V(M)) results in an S-SYNC-MINOR confirm. The SPM sets V(A) equal to the received serial number plus one.

7.22 *MAJOR SYNC POINT SPDU*

The MAJOR SYNC POINT SPDU is used to define a major synchronization point. A confirmation has to be received before more data can be sent on the normal and expedited flows. The right to issue a MAJOR SYNC POINT SPDU is subject to the token restrictions specified in Table 5/X.225.

7.22.1 *Content of MAJOR SYNC POINT SPDU*

The MAJOR SYNC POINT SPDU contains:

a) Sync Type Item parameter which is only present when indicating that this major synchronization point is not the end of the current activity;

b) Serial Number parameter which indicates the serial number of this major synchronization point, and is set by the SPM to the current value of V(M);

c) User Data parameter which allows a limited amount of transparent user data to be transferred.

7.22.2 *Sending the MAJOR SYNC POINT SPDU*

An S-SYNC-MAJOR request results in a MAJOR SYNC POINT SPDU. This SPDU is sent on the transport normal flow. If Vsc is true, V(A) is set equal to V(M) and Vsc is set false. V(M) is incremented by one. If the activity management functional unit has been selected, Vnextact is set true. If the transport expedited flow is available to this session connection, the SPM waits for a PREPARE (MAJOR SYNC ACK) SPDU, followed by a MAJOR SYNC ACK SPDU. Otherwise, just a MAJOR SYNC ACK is expected. Any other SPDUs received prior to the MAJOR SYNC ACK SPDU will result in the appropriate service indications being given to the SS-user.

7.22.3 *Receiving the MAJOR SYNC POINT SPDU*

A valid incoming MAJOR SYNC POINT SPDU (with received serial number equal to V(M)) results in an S-SYNC-MAJOR indication. If Vsc is false, V(A) is set equal to V(M). V(M) is incremented by one. If the activity management functional unit has been selected, Vnextact is set true.

7.23 *MAJOR SYNC ACK SPDU*

The MAJOR SYNC ACK SPDU is used to return a confirmation to a major synchronization point.

7.23.1 *Content of MAJOR SYNC ACK SPDU*

The MAJOR SYNC ACK SPDU contains:

a) Serial Number parameter which indicates the serial number of the major synchronization point which is being confirmed (which is equal to V(M) minus one);

b) User Data parameter which allows a limited amount of transparent user data to be transferred.

7.23.2 *Sending the MAJOR SYNC ACK SPDU*

An S-SYNC-MAJOR response results in a MAJOR SYNC ACK SPDU. This SPDU is sent on the transport normal flow. If the transport expedited flow is available to this session connection, a PREPARE (MAJOR SYNC ACK) SPDU is sent simultaneously, or earlier, on the transport expedited flow. V(A) and V(R) are set equal to V(M). If the activity management functional unit has been selected, Vact is set to Vnextact.

7.23.3 *Receiving the MAJOR SYNC ACK SPDU*

A valid incoming MAJOR SYNC ACK SPDU (with received serial number equal to V(M) minus one) results in an S-SYNC-MAJOR confirm.

If the transport expedited flow is available to this session connection, two successive SPDUs will be received:

a) PREPARE (MAJOR SYNC ACK) SPDU on the transport expedited flow, followed by

b) MAJOR SYNC ACK SPDU on the transport normal flow.

V(A) and V(R) are set equal to V(M). If the activity management functional unit has been selected, Vact is set to Vnextact.

7.24 *RESYNCHRONIZE SPDU*

The RESYNCHRONIZE SPDU is used to provide the SS-users with a selective means to resynchronize the exchange of data to a synchronization point and to reposition the tokens to an agreed side. Use of this procedure may result in loss of data.

This SPDU can also be used to "purge" the session connection, since that is a particular case of resynchronization. The following options are provided:

a) abandon;

b) set;

c) restart.

Since the resynchronization protocol provides a repositioning of the tokens a particular use of it is the destructive way to get the tokens.

When used with activity management, the RESYNCHRONIZE SPDU can only be sent when Vact is true.

7.24.1 *Content of RESYNCHRONIZE SPDU*

The RESYNCHRONIZE SPDU contains:

a) Token Setting Item which indicates the requestor's proposed token positions for all available tokens;

b) Resync Type Item parameter which indicates the resynchronize option (abandon, set or restart);

c) Serial Number parameter which indicates the serial number to which resyncrhonization is being requested. The serial number is supplied by the SS-user if the resynchronize option is set or restart. If the resynchronize option is abandon, the serial number is set to the value of V(M) of the sending SPM;

d) User Data parameter which allows a limited amount of transparent user data to be transferred.

7.24.2 *Sending the RESYNCHRONIZE SPDU*

An S-RESYNCHRONIZE request (with serial number greater than or equal to V(R) and less than or equal to V(M) if the resynchronize option is restart) results in a RESYNCHRONIZE SPDU. This SPDU is sent on the transport normal flow. If the transport expedited flow is available to this session connection, a PREPARE (RESYNCHRONIZE) SPDU is sent simultaneously, or earlier, on the transport expedited flow.

The SPM goes into a state where all the incoming SPDUs are discarded except PREPARE (RESYNCHRONIZE), RESYNCHRONIZE, PREPARE (RESYNCHRONIZE ACK), RESYNCHRONIZE ACK, ACTIVITY DISCARD, ACTIVITY INTERRUPT and ABORT SPDUs.

If a RESYNCHRONIZE, PREPARE (RESYNCHRONIZE), ACTIVITY INTERRUPT or ACTIVITY DISCARD SPDU is received when the SPM is in this state, a resynchronization contention situation has occurred and is dealt with as specified in § 7.24.4.

7.24.3 *Receiving the RESYNCHRONIZE SPDU*

Except when a resynchronization contention situation has occurred, a valid incoming RESYNCHRONIZE SPDU (with received serial number greater than or equal to V(R) if the resynchronize option is restart) results in an S-RESYNCHRONIZE indication. If the resynchronize option is abandon, this indication contains a serial number which is equal to V(M) or the received serial number, whichever is higher; V(M) is set to this value. If the transport expedited flow is available to this session connection, two successive SPDUs will be received:

a) PREPARE (RESYNCHRONIZE) SPDU on the transport expedited flow, followed by

b) RESYNCHRONIZE SPDU on the transport normal data flow.

When the PREPARE (RESYNCHRONIZE) SPDU is received, all subsequently received SPDUs, except ABORT SPDU, are discarded until the RESYNCHRONIZE SPDU is received on the transport normal flow.

The SPM now waits for an S-RESYNCHRONIZE response.

If a resynchronization contention situation has occurred, only the contention loser (see § 7.24.4) passes an S-RESYNCRONIZE indication to the SS-user.

7.24.4 *Resynchronization contention*

The contention between two RESYNCHRONIZE, ACTIVITY INTERRUPT, or ACTIVITY DISCARD SPDUs is resolved according to Table 9/X.225. The table defines the contention winner whose SPDU is taken into account; the other SPDU is discarded.

If an incoming RESYNCHRONIZE SPDU is not acceptable, the receiving SS-user may issue another if it prevails over the original proposal according to the decision rules.

7.25 *RESYNCHRONIZE ACK SPDU*

The RESYNCHRONIZE ACK SPDU is used to notify the sender of a RESYNCHRONIZE SPDU of the completion of resynchronization.

TABLE 9/X.225

Contention winner

Outgoing SPDU from SPM A \ Incoming SPDU from SPM B	RESYNC. (abandon)	RESYNC. (set)	RESYNC. (restart)	ACTIVITY INTERRUPT	ACTIVITY DISCARD
RESYNC. (abandon)	Initiator	SPM A	SPM A	SPM B	SPM B
RESYNC. (set)	SPM B	Initiator	SPM A	SPM B	SPM B
RESYNC. (restart)	SPM B	SPM B	SPM with lower serial number or if equal then initiator	SPM B	SPM B
ACTIVITY INTERRUPT	SPM A	SPM A	SPM A	See note	See note
ACTIVITY DISCARD	SPM A	SPM A	SPM A	See note	See note

Note – Collision is not possible in these cases because only the owner of the major/activity token is permitted to send ACTIVITY DISCARD SPDU or ACTIVITY INTERRUPT SPDU.

7.25.1 *Content of RESYNCHRONIZE ACK SPDU*

The RESYNCHRONIZE ACK SPDU contains:

a) Token Setting Item parameter which indicates the selected token positions.

b) Serial Number parameter which indicates the first serial number to be used in the resynchronized flow. This parameter is set according to the Resync Type Item parameter in the received RESYNCHRONIZE SPDU:

1) for the restart option, to the serial number in the received RESYNCHRONIZE SPDU;

2) for the set option, to the serial number in the S-RESYNCHRONIZE response;

3) for the abandon option, to V(M).

c) User Data parameter which allows a limited amount of transparent user data to be transferred.

7.25.2 *Sending the RESYNCHRONIZE ACK SPDU*

An S-RESYNCHRONIZE response results in a RESYNCHRONIZE ACK SPDU. This SPDU is sent on the transport normal flow. If the transport expedited flow is available to this session connection, a PREPARE (RESYNCHRONIZE ACK) SPDU is sent simultaneously, or earlier, on the transport expedited flow.

The tokens are set to the values proposed by the requestor. If the requestor has indicated “accepting SS-user’s choice” for a token, then the acceptor’s proposed value for that token is used. The selected token settings are returned in the Token Setting Item of the RESYNCHRONIZE ACK SPDU.

V(A) and V(M) are set to the serial number contained in the RESYNCHRONIZE ACK SPDU.

V(R) is unchanged if the Resync Type Item parameter in the received RESYNCHRONIZE SPDU indicated the restart option. Otherwise, V(R) is set to zero.

7.25.3 *Receiving the RESYNCHRONIZE ACK SPDU*

A valid incoming RESYNCHRONIZE ACK SPDU results in an S-RESYNCHRONIZE confirm. If the transport expedited flow is available to this session connection, two successive SPDUs will be received:

a) PREPARE (RESYNCHRONIZE ACK) SPDU on the transport expedited flow, followed by

b) RESYNCHRONIZE ACK on the transport normal flow.

The tokens are set to the positions specified in the RESYNCHRONIZE ACK SPDU.

V(A) and V(M) are set to the serial number contained in the RESYNCHRONIZE ACK SPDU.

V(R) is unchanged if the Resync Type Item parameter in the transmitted RESYNCHRONIZE SPDU indicated the restart option. Otherwise, V(R) is set to zero.

7.26 *PREPARE SPDU*

The PREPARE SPDU is only used when the transport expedited flow is available to this session connection. It notifies the imminent arrival of certain SPDUs and indicates to the receiving SPM that SPDUs received on the transport normal flow may be discarded under certain circumstances.

7.26.1 *Content of PREPARE SPDU*

The PREPARE SPDU contains a Prepare Type parameter which indicates which SPDU should be expected on the transport normal flow.

7.26.2 *Sending the PREPARE SPDU*

The PREPARE SPDU is sent before the associated SPDUs specified in Table 10/X.225 when the transport expedited flow is available to this session connection. Table 10/X.225 also specifies the value of the Prepare Type parameter.

TABLE 10/X.225

SPDUs associated with the PREPARE SPDU

Associated SPDU	Prepare Type
RESYNCHRONIZE SPDU	RESYNCHRONIZE
RESYNCHRONIZE ACK SPDU	RESYNCHRONIZE ACK
MAJOR SYNC ACK SPDU	MAJOR SYNC ACK
ACTIVITY INTERRUPT SPDU	RESYNCHRONIZE
ACTIVITY INTERRUPT ACK SPDU	RESYNCHRONIZE ACK
ACTIVITY DISCARD SPDU	RESYNCHRONIZE
ACTIVITY DISCARD ACK SPDU	RESYNCHRONIZE ACK
ACTIVITY END ACK SPDU	MAJOR SYNC ACK
ABORT SPDU	ABORT

The PREPARE SPDU is sent on the transport expedited flow (its associated SPDU being sent on the transport normal flow). The SPM goes to a state which is determined by the initial request.

7.26.3 *Receiving the PREPARE SPDU*

A valid incoming PREPARE SPDU results in the SPM entering a state where it is waiting for the associated SPDU on the transport normal flow. If the Prepare Type parameter indicates MAJOR SYNC ACK, any SPDUs received on the transport normal flow are processed normally. Otherwise, SPDUs received on the transport normal flow before the indicated SPDU are discarded. In any case, if an EXPEDITED DATA SPDU is received after a PREPARE SPDU, but before the associated SPDU on the transport normal flow, the S-EXPEDITED-DATA indication is not passed to the SS-user until the associated SPDU has been received and processed.

7.27 *EXCEPTION REPORT SPDU*

The EXCEPTION REPORT SPDU is used to report that a protocol error has been detected within the SPM. It can only be sent in the data transfer phase and subject to the token restrictions specified in Table 5/X.225.

7.27.1 *Content of EXCEPTION REPORT SPDU*

The EXCEPTION REPORT SPDU contains a Reflect Parameter Values parameter which is used to indicate a field of arbitrary length, which contains the bit pattern of the SPDU received with a protocol error, up to and including the detected error.

7.27.2 *Sending the EXCEPTION REPORT SPDU*

On detection of a protocol error, for example an SPDU received at an unexpected time, or an invalid SPDU, the SPM may generate an EXCEPTION REPORT SPDU. This SPDU is sent on the transport normal flow. At the same time an S-P-EXCEPTION-REPORT indication will be generated. The SPM enters an error state which is only left when any of the following SPDUs, or their associated local service requests, are received:

ACTIVITY DISCARD;

ACTIVITY INTERRUPT;

RESYNCHRONIZE;

ABORT;

GIVE TOKENS (with the data token);

PREPARE (RESYNCHRONIZE).

Any other SPDUs received will be discarded. However, V(A) and V(M) will be updated appropriately if valid MINOR SYNC POINT SPDUs or MAJOR SYNC POINT SPDUs are received.

7.27.3 *Receiving the EXCEPTION REPORT SPDU*

When an incoming EXCEPTION REPORT SPDU is received, an S-P-EXCEPTION-REPORT indication is given and the SPM enters an error state.

The SPM leaves the error state when any of the following SPDUs, or their associated local service requests, are received:

ACTIVITY DISCARD;

ACTIVITY INTERRUPT;

RESYNCHRONIZE;

ABORT;

GIVE TOKENS (with the data token);

PREPARE (RESYNCHRONIZE).

Note – This action is dependent on the receipt of the EXCEPTION REPORT SPDU, not on examination of its parameter value. This enables the procedure to be followed in cases where the implementation cannot deal with an SPDU length greater than the minimum specified in § 8.3.27.3.

7.28 *EXCEPTION DATA SPDU*

The EXCEPTION DATA SPDU is used to put the SPM into an error state.

It can only be sent subject to the token restrictions specified in Table 5/X.225 and:

a) when the activity management functional unit has been selected and an activity is in progress, or

b) the activity management functional unit has not been selected.

7.28.1 *Content of EXCEPTION DATA SPDU*

The EXCEPTION DATA SPDU contains:

a) Reason Code parameter which indicates the reason for sending the EXCEPTION DATA SPDU;

b) User Data parameter which allows a limited amount of transparent user data to be transferred.

7.28.2 *Sending the EXCEPTION DATA SPDU*

An S-U-EXCEPTION-REPORT request results in the SPM sending an EXCEPTION DATA SPDU on the transport normal flow. The SPM enters an error state. The error state will be left when an S-U-ABORT request or a T-DISCONNECT indication is received or when any of the following SPDUs are received:

ACTIVITY DISCARD;

ACTIVITY INTERRUPT;

ABORT;

RESYNCHRONIZE;

GIVE TOKENS (with the data token);

PREPARE (RESYNCHRONIZE).

Any other SPDUs received will be discarded. However, V(A) and V(M) will be updated appropriately if MINOR SYNC POINT SPDUs or MAJOR SYNC POINT SPDUs are received.

7.28.3 *Receiving the EXCEPTION DATA SPDU*

A valid incoming EXCEPTION DATA SPDU results in an S-U-EXCEPTION-REPORT indication. The SPM enters an error state, unless the data token is not assigned to this SPM, in which case the SPM state is unchanged.

The SPM leaves the error state when any of the following service primitives are invoked by the SS-user:

S-U-ABORT request;

S-RESYNCHRONIZE request;

S-ACTIVITY-DISCARD request;

S-ACTIVITY-INTERRUPT request;

S-TOKEN-GIVE request (with the data token).

7.29 *ACTIVITY START SPDU*

The ACTIVITY START SPDU is used to notify the beginning of an activity. The right to issue an ACTIVITY START SPDU is subject to the token restrictions specified in Table 5/X.225.

7.29.1 *Content of ACTIVITY START SPDU*

The ACTIVITY START SPDU contains:

a) Activity Identifier parameter which allows the SS-users to identify the activity being started;

b) User Data parameter which allows a limited amount of transparent user data to be transferred.

7.29.2 *Sending the ACTIVITY START SPDU*

An S-ACTIVITY-START request (when Vact is false) results in an ACTIVITY START SPDU. V(A), V(M) and V(R) are set to one. Vact is set true. This SPDU is sent on the transport normal flow.

7.29.3 *Receiving the ACTIVITY START SPDU*

A valid incoming ACTIVITY START SPDU (when Vact is false) results in an S-ACTIVITY-START indication. V(A), V(M) and V(R) are set to one. Vact is set true.

7.30 *ACTIVITY RESUME SPDU*

The ACTIVITY RESUME SPDU is used to notify the resumption of a previously interrupted activity. The right to issue an ACTIVITY RESUME SPDU is subject to the token restrictions specified in Table 5/X.225.

7.30.1 *Content of ACTIVITY RESUME SPDU*

The ACTIVITY RESUME SPDU contains:

a) Linking Information parameter group which contains:
   1) Called SS-user Reference parameter;
   2) Calling SS-user Reference parameter;
   3) Common Reference parameter;
   4) Additional Reference Information parameter;
   5) Old Activity Identifier which enables the SS-users to identify the old activity which is being resumed;
   6) Serial Number parameter which indicates the first serial number to be used minus one.

b) New Activity Identifier parameter which allows the SS-users to assign a new identifier to the activity being resumed.

c) User Data parameter which allows a limited amount of transparent user data to be transferred.

7.30.2 *Sending the ACTIVITY RESUME SPDU*

An S-ACTIVITY-RESUME request (when Vact is false) results in an ACTIVITY RESUME SPDU. V(A) and V(M) are set to the serial number provided by the SS-user plus one. V(R) is set to one. Vact is set true. This SPDU is sent on the transport normal flow.

7.30.3 *Receiving the ACTIVITY RESUME SPDU*

A valid incoming ACTIVITY RESUME SPDU (when Vact is false) results in an S-ACTIVITY-RESUME indication. V(A) and V(M) are set to the received serial number plus one. V(R) is set to one. Vact is set true.

7.31 *ACTIVITY INTERRUPT SPDU*

The ACTIVITY INTERRUPT SPDU is used to notify the interruption of an ongoing activity. The right to issue an ACTIVITY INTERRUPT SPDU is subject to the token restrictions specified in Table 5/X.225. Use of this procedure may result in loss of data.

7.31.1 *Content of ACTIVITY INTERRUPT SPDU*

The ACTIVITY INTERRUPT SPDU contains:

a) a Reason Code parameter which indicates the reason for sending the ACTIVITY INTERRUPT SPDU;

b) a User Data parameter which allows a limited amount of transparent user data to be transferred. This parameter shall not be present if protocol version 1 is selected.

7.31.2 *Sending the ACTIVITY INTERRUPT SPDU*

An S-ACTIVITY-INTERRUPT request results in an ACTIVITY INTERRUPT SPDU. This SPDU is sent on the transport normal flow. If the transport expedited flow is available to this session connection, a PREPARE (RESYNCHRONIZE) SPDU is sent simultaneously, or earlier, on the transport expedited flow. The SPM goes into a state where all incoming SPDUs are discarded except PREPARE (RESYNCHRONIZE ACK), ACTIVITY INTERUPT ACK and ABORT SPDUs.

7.31.3 *Receiving the ACTIVITY INTERRUPT SPDU*

A valid incoming ACTIVITY INTERRUPT SPDU results in an S-ACTIVITY-INTERRUPT indication. If the transport expedited flow is available to this session connection, two successive SPDUs will be received:

a) PREPARE (RESYNCHRONIZE) SPDU (see § 7.24) on the transport expedited flow, followed by

b) ACTIVITY INTERRUPT SPDU on the transport normal flow.

The SPM now waits for an S-ACTIVITY-INTERRUPT response.

7.32 *ACTIVITY INTERRUPT ACK SPDU*

The ACTIVITY INTERRUPT ACK SPDU is used to notify the sender of an ACTIVITY INTERRUPT SPDU of the completion of the interruption of the ongoing activity. On completion, all available tokens are assigned to the sender of the ACTIVITY INTERRUPT SPDU.

7.32.1 *Content of ACTIVITY INTERRUPT ACK SPDU*

The ACTIVITY INTERRUPT ACK SPDU contains a User Data parameter which allows a limited amount of transparent user data to be transfered. This parameter shall not be present if protocol version 1 is selected.

7.32.2 *Sending the ACTIVITY INTERRUPT ACK SPDU*

An S-ACTIVITY-INTERRUPT response results in an ACTIVITY INTERRUPT ACK SPDU. This SPDU is sent on the transport normal flow. If the transport expedited flow is available to this session connection, a PREPARE (RESYNCHRONIZE ACK) SPDU is sent simultaneously, or earlier, on the transport expedited flow. Vact is set false when the ACTIVITY INTERRUPT ACK SPDU has been sent.

7.32.3 *Receiving the ACTIVITY INTERRUPT ACK SPDU*

A valid incoming ACTIVITY INTERRUPT ACK SPDU results in an S-ACTIVITY-INTERRUPT confirm. If the transport expedited flow is available to this session connection, two successive SPDUs will be received:

a) PREPARE (RESYNCHRONIZE ACK) SPDU (see § 7.25) on the transport expedited flow, followed by

b) ACTIVITY INTERRUPT ACK SPDU on the transport normal flow.

Vact is set false when the ACTIVITY INTERRUPT ACK SPDU has been received.

7.33 *ACTIVITY DISCARD SPDU*

The ACTIVITY DISCARD SPDU is used to notify the cancellation of an ongoing activity. The right to issue an ACTIVITY DISCARD SPDU is subject to the token restrictions specified in Table 5/X.225. Use of this procedure may result in the loss of data.

7.33.1 *Content of ACTIVITY DISCARD SPDU*

The ACTIVITY DISCARD SPDU contains:

a) a Reason Code parameter which indicates the reason for sending the ACTIVITY DISCARD SPDU,

b) a User Data parameter which allows a limited amount of transparent user data to be transferred. This parameter shall not be present if protocol version 1 is selected.

7.33.2 *Sending the ACTIVITY DISCARD SPDU*

An S-ACTIVITY-DISCARD request results in an ACTIVITY DISCARD SPDU. This SPDU is sent on the transport normal flow. If the transport expedited flow is available to this session connection, a PREPARE (RESYNCHRONIZE) SPDU is sent simultaneously, or earlier, on the transport expedited flow. The SPM goes into a state where all the incoming SPDUs are discarded except PREPARE (RESYNCHRONIZE ACK), ACTIVITY DISCARD ACK and ABORT SPDUs.

7.33.3 *Receiving the ACTIVITY DISCARD SPDU*

A valid incoming ACTIVITY DISCARD SPDU results in an S-ACTIVITY-DISCARD indication. If the transport expedited flow is available to this session connection, two successive SPDUs will be received:

a) PREPARE (RESYNCHRONIZE) SPDU (see § 7.24) on the transport expedited flow, followed by

b) ACTIVITY DISCARD SPDU on the transport normal flow.

The SPM now waits for an S-ACTIVITY-DISCARD response.

7.34 *ACTIVITY DISCARD ACK SPDU*

The ACTIVITY DISCARD ACK SPDU is used to notify the sender of an ACTIVITY DISCARD SPDU of the completion of the cancellation of the ongoing activity. On completion, all available tokens are assigned to the sender of the ACTIVITY DISCARD SPDU.

7.34.1 *Content of ACTIVITY DISCARD ACK SPDU*

The ACTIVITY DISCARD ACK SPDU contains a User Data parameter which allows a limited amount of transparent user data to be transfered. This parameter shall not be present if protocol version 1 is selected.

7.34.2 *Sending the ACTIVITY DISCARD ACK SPDU*

An S-ACTIVITY-DISCARD response results in an ACTIVITY DISCARD ACK SPDU. This SPDU is sent on the transport normal flow. If the transport expedited flow is available to this session connection, a PREPARE (RESYNCHRONIZE ACK) SPDU is sent simultaneously, or earlier, on the transport expedited flow. Vact is set false when the ACTIVITY DISCARD ACK SPDU has been sent.

7.34.3 *Receiving the ACTIVITY DISCARD ACK SPDU*

A valid incoming ACTIVITY DISCARD ACK SPDU results in an S-ACTIVITY-DISCARD confirm. If the transport expedited flow is available to this session connection, two successive SPDUs will be received:

a) PREPARE (RESYNCHRONIZE ACK) SPDU (see § 7.25) on the transport expedited flow, followed by

b) ACTIVITY DISCARD ACK SPDU on the transport normal flow.

Vact is set false when the ACTIVITY DISCARD ACK SPDU has been received.

7.35 *ACTIVITY END SPDU*

The ACTIVITY END SPDU is used to define an implicit major synchronization point at the end of an activity. A confirmation has to be received before more data can be sent on the normal and expedited flows. The right to issue an ACTIVITY END SPDU is subject to the token restrictions specified in Table 5/X.225.

An ACTIVITY END SPDU can only be validly sent when Vact is true.

7.35.1 *Content of ACTIVITY END SPDU*

The ACTIVITY END SPDU contains:

a) Serial Number parameter which indicates the serial number of this major synchronization point, and is set by the SPM to the current value of V(M);

b) User Data parameter which allows a limited amount of transparent user data to be transferred.

7.35.2 *Sending the ACTIVITY END SPDU*

An S-ACITIVITY-END request (when Vact is true) results in an ACTIVITY END SPDU. This SPDU is sent on the transport normal flow. If Vsc is true, V(A) is set equal to V(M) and Vsc is set false. V(M) is incremented by one. Vnextact is set false. If the transport expedited flow is available to this session connection, the SPM waits for a PREPARE (MAJOR SYNC ACK) SPDU, followed by an ACTIVITY END ACK SPDU. Otherwise, just an ACTIVITY END ACK SPDU is expected. Any other SPDUs received prior to the ACTIVITY END ACK SPDU will result in the appropriate service indications being given to the SS-user.

7.35.3 *Receiving the ACTIVITY END SPDU*

A valid incoming ACTIVITY END SPDU (when Vact is true, and the received serial number equals V(M)) results in an S-ACTIVITY-END indication. If Vsc is false, V(A) is set equal to V(M). V(M) is incremented by one. Vnextact is set false.

7.36 *ACTIVITY END ACK SPDU*

The ACTIVITY END ACK SPDU is used to return a confirmation to an ACTIVITY END SPDU.

7.36.1 *Content of ACTIVITY END ACK SPDU*

The ACTIVITY END ACK SPDU contains:

a) Serial Number parameter which indicates the serial number of the major synchronization point which is being confirmed (which is equal to V(M) minus one);

b) User Data parameter which allows a limited amount of transparent user data to be transferred.

7.36.2 *Sending the ACTIVITY END ACK SPDU*

An S-ACTIVITY-END response results in an ACTIVITY END ACK SPDU. This SPDU is sent on the transport normal flow. If the transport expedited flow is available to this session connection, a PREPARE (MAJOR SYNC ACK) SPDU is sent simultaneously, or earlier, on the transport expedited flow. V(A) and V(R) are set equal to V(M). Vact is set to Vnextact.

7.36.3 *Receiving the ACTIVITY END ACK SPDU*

A valid incoming ACTIVITY END ACK SPDU (with Vsc false and received serial number equal to V(M) minus one) results in an S-ACTIVITY-END confirm.

If the transport expedited flow is available to this session connection, two successive SPDUs will be received:

a) PREPARE (MAJOR SYNC ACK) SPDU on the transport expedited flow, followed by

b) ACTIVITY END ACK SPDU on the transport normal flow.

V(A) and V(R) are set equal to V(M). Vact is set to Vnextact.

7.37 *Additional elements of procedure for segmented SSDUs*

The following SPDUs may contain segments of the associated SSDU:

ACCEPT SPDU
REFUSE SPDU
FINISH SPDU
DISCONNECT SPDU
NOT FINISHED SPDU
CAPABILITY DATA SPDU
CAPABILITY DATA ACK SPDU
GIVE TOKENS SPDU
PLEASE TOKENS SPDU
GIVE TOKENS CONFIRM SPDU
MINOR SYNC POINT SPDU
MINOR SYNC ACK SPDU
MAJOR SYNC POINT SPDU
MAJOR SYNC ACK SPDU
RESYNCHRONIZE SPDU
RESYNCHRONIZE ACK SPDU
EXCEPTION DATA SPDU
ACTIVITY START SPDU
ACTIVITY RESUME SPDU
ACTIVITY INTERRUPT SPDU
ACTIVITY INTERRUPT ACK SPDU
ACTIVITY DISCARD SPDU
ACTIVITY DISCARD ACK SPDU
ACTIVITY END SPDU
ACTIVITY END ACK SPDU

These SPDUs are subject to the following additional procedures.

7.37.1 *Content of the SPDU*

Where an SSDU is segmented, the first SPDU contains all the parameters which would have been present in the SPDU if the SSDU had not been segmented, together with an Enclosure Item parameter, which indicates beginning of SSDU and not end of SSDU, and at least one octet of User Data. The last SPDU of the SSDU contains the Enclosure Item parameter which indicates not beginning of SSDU and end of SSDU and may or may not contain User Data. Intermediate SPDUs of the SSDU, if present, contain the Enclosure Item parameter which indicates not beginning of SSDU and not end of SSDU and at least one octet of User Data.

7.37.2 *Sending the SPDU*

The sending procedures for SPDUs where these additional elements of procedure apply are extended in the following way:

a) where the SPM sends an SPDU, it shall send an ordered sequence of SPDUs which together comprise the complete SSDU;

b) sending this ordered sequence of SPDUs shall be interrupted when the SPM sends an ABORT SPDU or a PREPARE (ABORT) SPDU (for example, as a result of an S-U-ABORT request or a detected protocol error) or when the SPM receives an ABORT SPDU, a PREPARE (ABORT) SPDU or a T-DISCONNECT indication. In this case, the SPM shall stop sending the ordered sequence of SPDUs and shall take the appropriate defined actions;

Note – the ordered sequence of SPDUs sent so far will not comprise a complete SSDU. The Enclosure Item parameter will not have been sent with a value which indicates end of SSDU.

c) as a local matter, sending this ordered sequence of SPDUs may be interrupted when the SPM receives an SPDU which will cause the ordered sequence of SPDUs to be discarded by the remote SPM. In this situation, the SPM which is sending the ordered sequence of SPDUs is not required to send the remainder of the ordered sequence.

Note – This situation will occur if the received destructive SPDU was sent by the remote SPM before the first SPDU of the ordered sequence of SPDUs was received by the remote SPM, or if the remote SPM took the local implementation decision indicated in § 7.37.3 d).

7.37.3 *Receiving the SPDU*

The receiving procedures for SPDUs where these additional elements of procedure apply are extended in the following way:

a) where the SPM receives an SPDU, it shall receive an ordered sequence of SPDUs which together comprise the complete SSDU;

b) receiving this ordered sequence of SPDUs shall be interrupted when the SPM receives an ABORT SPDU, a PREPARE (ABORT) SPDU or a T-DISCONNECT indication. This shall have a destructive effect on the entire segmented SSDU (i.e. the SPDUs which have already been received are discarded). The SPM shall take the appropriate defined actions;

c) the SPM may send an ABORT SPDU or a PREPARE (ABORT) SPDU (for example, as a result of an S-U-ABORT request or a detected protocol error) while receiving an ordered sequence of SPDUs. This shall have a destructive effect on the entire segmented SSDU being received (i.e. SPDUs comprising part of the segmented SSDU which have already been received are discarded and any SPDUs comprising part of the segmented SSDU which are received subsequently are discarded). The SPM shall take the appropriate defined actions;

d) as a local matter, while receiving this ordered sequence of SPDUs, the SPM may send any other appropriate SPDU which will have a destructive effect on the entire SSDU being received (i.e. SPDUs comprising part of the segmented SSDU which have already been received will be discarded and any SPDUs comprising part of the segmented SSDU which are received subsequently will be discarded).

Note – The conditions and effect stated here are as though the segmented SSDU had been sent in a single SPDU and the SPDU causing the destructive effect had been sent before that SPDU is received.

8 Structure and encoding of SPDUs

8.1 *TSDU structure*

Each TSDU consists of one or more SPDUs complying with the requirements for concatenation (see § 6.3.7).

Each SPDU within a TSDU consists of one or more octets that are numbered sequentially starting from 1.

Each octet within an SPDU consists of eight bits numbered 8 to 1 where 1 is the low ordered bit.

The sequence of octets within an SPDU and the sequence of bits within an octet are defined for each SPDU in § 8.3, with the additional convention that where the text refers to bits within a two-octet field and the bits are numbered 16 to 1, then 1 is the low order bit and the octet containing bits 16 to 9 precedes the octet containing bits 8 to 1 in the SPDU.

Within each TSDU:

a) the sequential ordering of SPDUs is maintained;

b) the ordering of the octets is maintained in the same order as in the SPDU;

c) the ordering of bits within each TSDU is maintained in the same order as in the SPDU (i.e. the low order bit is mapped onto the low order bit and the high order bit is mapped onto the high order bit).

Note 1 – The TSDU structure is illustrated in Figure 4/X.225. The integrity of this structure is maintained over a transport connection. This Recommendation does not define the way in which the TSDU is transmitted.

Note 2 – When the structure of an SPDU is illustrated in this Recommendation, the following convention is used:

a) octets are shown with the lowest numbered octet to the left, higher numbered octets being shown further to the right;

b) within an octet, bits are shown with bit 8 to the left and bit 1 to the right.

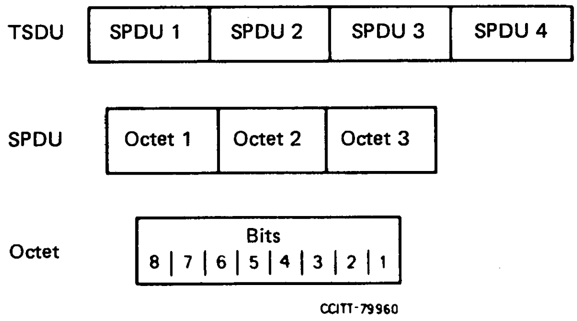


FIGURE 4/X.225

Illustration of definition of TSDU structure

8.2 *SPDU structure*

This subsection specifies the general structure of SPDUs in terms of their constituent fields. This structure is illustrated in Figure 5/X.225.

Codings and structural requirements specific to particular SPDUs are specified in § 8.3.

Examples of valid SPDU structure are illustrated in Figure 6/X.225.

8.2.1 *SPDUs*

SPDUs shall contain, in the following order:

a) *the SI field* that identifies the type of SPDU (see Note);

b) *the LI field* that indicates the length of the associated parameter field defined in § 8.2.1 c);

c) *the parameter field* which, if present, consists of the PGI units (see § 8.2.2) and/or PI units (see § 8.2.3) defined for the SPDU;

d) *the user information field*, if defined for the SPDU and if present.

Note – The SI field encompasses both the CI field and the RI field defined in Recommendation T.62. The protocol specified in this Recommendation does not require a distinction to be made between these two fields.

8.2.2 *PGI units*

PGI units shall contain, in the following order:

a) *the PGI field* that identifies the parameter group;

b) *the LI field* that indicates the length of the associated parameter field defined in § 8.2.2 c);

c) *the parameter field* which, if present, consists of either:

1) a single parameter value (see Note); or
2) one or more PI units (see § 8.2.3).

Note – A PGI unit with one parameter is structurally equivalent to a PI unit, but the distinction has been retained in order to maintain compatibility with Recommendation T.62.

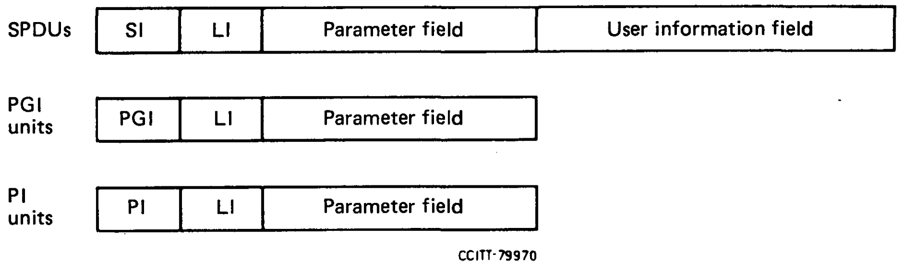
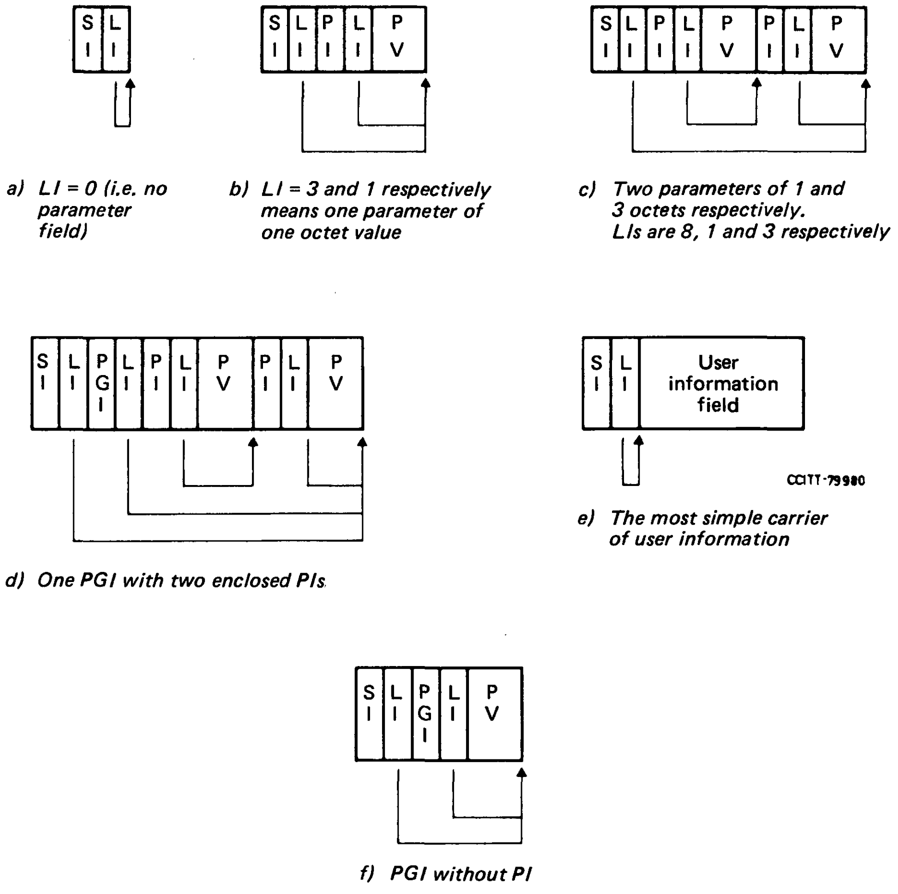


FIGURE 5/X.225

Illustration of structure of SPDUs, PGI and PI units



a) *LI = 0 (i.e. no parameter field)*

b) *LI = 3 and 1 respectively means one parameter of one octet value*

c) *Two parameters of 1 and 3 octets respectively. LIs are 8, 1 and 3 respectively*

d) *One PGI with two enclosed PIs*

e) *The most simple carrier of user information*

f) *PGI without PI*

FIGURE 6/X.225

Examples of SPDU structure

8.2.3 *PI units*

PI units shall contain, in the following order:

a) the *PI field* that identifies the parameter;

b) the *LI field* that indicates the length of the associated parameter field defined in § 8.2.3 c);

c) the *parameter field* which, if present, consists of the parameter value.

8.2.4 *Identifier fields*

The SI field shall comprise one octet. The value of the SI field, specified as a decimal number in § 8.3, shall be encoded as a binary number.

The PGI and PI fields shall each comprise one octet and shall contain a PGI or PI code respectively. The PGI and PI codes are expressed as decimal numbers in the tables in § 8.3 and shall be encoded as a binary number.

8.2.5 *Length indicator field*

The value of the LI field is expressed as a binary number representing the length, in octets, of the associated parameter field (see Note). A value of zero indicates that the associated parameter field is absent.

LI fields indicating lengths within the range 0-254 shall comprise one octet.

LI fields indicating lengths within the range 255-65 535 shall comprise three octets. The first octet shall be coded 1111 1111 and the second and third octets shall contain the length of the associated parameter field with the high order bits in the first of these two octets.

Note – The value of the LI field does not include either itself or any subsequent user information.

8.2.6 *Parameter fields*

PGI units and PI units defined as mandatory in the tables in § 8.3 shall contain a parameter field of one or more octets.

Any PGI unit or PI unit defined as non-mandatory in the tables in § 8.3 may be omitted if it is not required for conveying information (i.e. a parameter value). If a PGI unit or a PI unit contains an LI field with the value zero, the associated parameter field is absent (see Note) and the value of the parameter field shall be considered as its default value.

Note – It is recommended that if a non-mandatory parameter is absent, the associated PGI (or PI) and LI fields should not be included in the SPDU.

PGI units and PI units within the same nesting level shall be ordered in increasing value of their PGI and PI codes.

PGI or PI units containing:

a) a PGI or PI code listed in Annex C,

b) a PGI or PI code not listed in § 8.3 or in Annex C,

are defined as valid.

Note – See § A.4.3 for actions to be taken by the SPM on receipt of SPDUs containing these PGI or PI units.

8.2.7 *Parameter values*

Bits within a parameter field which are indicated as reserved shall have those bits set to zero in the SPDU.

Note – See § A.4.3 for actions to be taken by the SPM on receipt of SPDUs containing such bits.

8.2.8 *User information fields*

Segments of a segmented SSDU shall be contained in the User Information Fields of SPDUs such that the order of the segments is maintained. An SSDU which is not segmented shall be contained in the User Information Field of a single SPDU. The order of the octets and the order of the bits in the SSDU shall be maintained in the SPDUs.

8.3 *SPDU identifiers and associated parameter fields*

The SPDUs specified in the remainder of this subsection do not, with certain exceptions, consider the case where an SSDU is segmented. When protocol version 2 is selected, most SSDUs may be segmented. (The circumstances under which an SSDU may be segmented are specified in § 6.3.5.) The additional encoding requirements when an SSDU is segmented are specified in § 8.4.

8.3.1 *CONNECT (CN) SPDU*

8.3.1.1 The SI field shall contain the value 13.

8.3.1.2 The parameter fields shall be as specified in Table 11/X.225.

8.3.1.3 The Calling SS-user Reference PV field shall be as defined by the calling SS-user.

8.3.1.4 The Common Reference PV field shall be as defined by the calling SS-user.

8.3.1.5 The Additional Reference Information PV field shall be as defined by the calling SS-user.

8.3.1.6 If the Connect/Accept Item is absent, the default values defined for the enclosed PI units shall apply.

8.3.1.7 The Protocol Options PV field shall indicate whether or not the initiator is able to receive extended concatenated SPDUs (see § 6.3.7). The encoding for this field shall be:

a) bit 1 = 1 : able to receive extended concatenated SPDUs;

b) bit 1 = 0 : not able to receive extended concatenated SPDUs.

Bits 2-8 are reserved.

If the Protocol Options PI unit or PV field is absent, SPDUs with extended concatenation cannot be received.

8.3.1.8 The TSDU Maximum Size PV shall be present if a TSDU Minimum Size PV is proposed:

a) the first two octets of the PV field shall contain the proposed maximum TSDU size, expressed in octets, in the direction from the initiator to the responder, encoded as a binary number, where the first of the two octets is the high order part of the number;

b) the second two octets of the PV field shall contain the proposed maximum TSDU size, expressed in octets, in the direction from the responder to the initiator, encoded as a binary number, where the first of the two octets is the high order part of the number.

If this parameter is not present, the TSDU Maximum Size is not limited over the session connection. If either pair of octets has the value zero, the TSDU size is not limited in the direction of transfer associated with that pair of octets.

8.3.1.9 The bits in the Version Number PV field shall indicate which protocol versions are proposed for use over the session connection:

a) bit 1 protocol version 1;

b) bit 1 protocol version 2.

Bits 3-8 are reserved.

The encoding for each bit shall be:

c) 0 : use of the protocol version not proposed;

d) 1 : use of the protocol version proposed.

If this PI unit or PV field is absent, the default shall be protocol version 1.

8.3.1.10 The Initial Serial Number PV field shall be present if the activity management functional unit is not proposed and any of the minor synchronize, major synchronize or resynchronize functional units are proposed. As an SS-user option, an Initial Serial Number PV field may be present if the activity management functional unit is proposed provided that any of the minor synchronize, major synchronize or resynchronize functional units are also proposed.

TABLE 11/X.225

Parameters of the CONNECT SPDU

PGI	m/nm	Code	PI	m/nm	Code	Length	Reference
Connection identifier	nm	1	Calling SS-user reference	nm	10	64 octets maximum	§ 7.1.1 a) 1) § 8.3.1.3
			Common reference	nm	11	64 octets maximum	§ 7.1.1 a) 2) § 8.3.1.4
			Additional reference information	nm	12	4 octets maximum	§ 7.1.1 a) 3) § 8.3.1.5
Connect/Accept item (see § 8.3.1.6)	nm	5	Protocol options	m	19	1 octet	§ 7.1.1 b) 1) § 8.3.1.7
			TSDU maximum size	nm	21	4 octets	§ 7.1.1 b) 2) § 8.3.1.8
			Version number	m	22	1 octet	§ 7.1.1 b) 3) § 8.3.1.9
			Initial serial number	nm	23	6 octets maximum	§ 7.1.1 b) 4) § 8.3.1.10
			Token setting item	nm	26	1 octet	§ 7.1.1 b) 5) § 8.3.1.11
			Session user requirements	nm	20	2 octets	§ 7.1.1 c) § 8.3.1.12
			Calling session selector	nm	51	16 octets maximum	§ 7.1.1 d) § 8.3.1.13
			Called session selector	nm	52	16 octets maximum	§ 7.1.1 d) § 8.3.1.14
User data	nm	193				512 octets maximum	§ 7.1.1 e) § 8.3.1.15
			Data overflow	nm	60	1 octet	§ 7.1.1 f) § 8.3.1.17
Extended user data	nm	194				10 240 octets maximum	§ 7.1.1 g) § 8.3.1.16

m: mandatory.
nm: not mandatory (see § 8.2.6).

Each digit of the serial number is encoded as an octet, as follows:

a) 0 : 0011 0000;

b) 1 : 0011 0001;

c) 2 : 0011 0010;

d) 3 : 0011 0011;

e) 4 : 0011 0100;

f) 5 : 0011 0101;

g) 6 : 0011 0110;

h) 7 : 0011 0111;

i) 8 : 0011 1000;

j) 9 : 0011 1001.

Serial number can range from 0 to 999 999. The most significant digit is encoded first in the PV field. Leading zeros may be omitted.

8.3.1.11 The Token Setting Item PV field, if present, shall indicate the initial position of the tokens. The bits of the Token Setting Item PV field are defined as bit pairs:

a) bits 8, 7 release token;

b) bits 6, 5 major/activity token;

c) bits 4, 3 synchronize-minor token;

d) bits 2, 1 data token.

The encoding for each bit pair shall be:

e) 00 : initiator's side;

f) 01 : responder's side;

g) 10 : called SS user's choice;

h) 11 : reserved.

The values are relevant only if the appropriate functional units are requested in the Session User Requirements parameter. If no functional unit requiring a token has been requested, this parameter need not be present.

If this PI unit or PV field is absent, the default shall be that all tokens whose availability is proposed in the Session User Requirements parameter are assigned to the calling SS-user.

8.3.1.12 The bits in the Session User Requirements PV field shall indicate the functional units proposed by the calling SS-user, for use over this session connection:

a) bit 1 half-duplex functional unit;

b) bit 2 duplex functional unit;

c) bit 3 expedited data functional unit;

d) bit 4 minor synchronize functional unit;

e) bit 5 major synchronize functional unit;

f) bit 6 resynchronize functional unit;

g) bit 7 activity management functional unit;

h) bit 8 negotiated release functional unit;

i) bit 9 capability data functional unit;

j) bit 10 exceptions functional unit;

k) bit 11 typed data functional unit.

Bits 12-16 are reserved.

When this parameter is present, at least one of the half-duplex and the duplex functional units shall be proposed.

The encoding for each bit shall be:

l) 0 : use of functional unit not proposed;

m) 1 : use of functional unit proposed.

When this parameter is absent, the default shall be as though bits 1, 4, 7, 9 and 10 are set to one and the remaining bits are set to zero.

8.3.1.13 The Calling Session Selector, if present, shall be derived from the Calling Session Address supplied by the calling SS-user. When this parameter is absent, the default shall be as though this parameter was set to a null value.

8.3.1.14 The Called Session Selector, if present, shall be derived from the Called Session Address supplied by the calling SS-user. When this parameter is absent, the default shall be as though this parameter was set to a null value.

8.3.1.15 The User Data PV field, if present, shall contain user data supplied by the calling SS-user.

8.3.1.16 The Extended User Data parameter, if present, shall contain user data supplied by the calling SS-user. This parameter shall be present if the Data Overflow parameter is present. This parameter shall not be present if protocol version 1 is proposed.

Only one of the User Data and Extended User Data parameters may be present (see § 7.1.1).

8.3.1.17 The Data Overflow parameter, if present, shall indicate that there is more than 10 240 octets of user data to be transformed. This parameter shall not be present if protocol version 1 is proposed.

The encoding of this field shall be:

bit 1 = 1 more than 10 240 octets of user data;

bit 1 shall never be set equal to 0.

Bits 2-8 are reserved.

If the Data Overflow PI unit or PV field is absent, there are not more than 10 240 octets of user data.

8.3.2 *OVERFLOW ACCEPT (OA) SPDU*

8.3.2.1 The SI field shall contain the value 16.

8.3.2.2 The parameter fields shall be as specified in Table 12/X.225.

TABLE 12/X.225

Parameters of the OVERFLOW ACCEPT SPDU

PGI	m/nm	Code	PI	m/nm	Code	Length	Reference
			TSDU maximum size	nm	21	4 octets	§ 7.2.1 a) § 8.3.2.3
			Version number	m	22	1 octet	§ 7.2.1 b) § 8.3.2.4

m: mandatory.
nm: not mandatory (see § 8.2.6).

8.3.2.3 The TSDU Maximum Size parameter shall be present if a TSDU maximum size is proposed by the receiver. The encoding and default for this field is defined in § 8.3.1.8.

8.3.2.4 In the Version Number PV field bit 2 shall have the value 1 indicating that protocol version 2 is proposed (and selected) for use over this session connection. Bit 1 shall have the value 0 indicating that protocol version 1 is not proposed.

Bits 3-8 are reserved.

8.3.3 *CONNECT DATA OVERFLOW (CDO) SPDU*

8.3.3.1 The SI field shall contain the value 15.

8.3.3.2 The parameter fields shall be as specified in Table 13/X.225.

TABLE 13/X.225

Parameters of the CONNECT DATA OVERFLOW

PGI	m/nm	Code	PI	m/nm	Code	Length	Reference
			Enclosure item	m	25	1 octet	§ 7.3.1 a) § 8.3.3.3
User data	nm	193				65 528 octets maximum	§ 7.3.1 b) § 8.3.3.4

m: mandatory.
nm: not mandatory (see § 8.2.6).

8.3.3.3 The Enclosure Item PV field shall indicate whether or not this SPDU is the end of the SSDU.

The encoding for this field shall be:

a) bit 1 = 0 not beginning of SSDU;

b) bit 2 = 1 end of SSDU;
bit 2 = 0 not end of SSDU.
Bits 3-8 are reserved.

8.3.3.4 The User Data field, if present, shall contain a segment of the associated SSDU. The User Data field shall be present if the Enclosure Item has bit 2 = 0.

8.3.4 *ACCEPT (AC) SPDU*

8.3.4.1 The SI field shall contain the value 14.

8.3.4.2 The parameter fields shall be as specified in Table 14/X.225.

8.3.4.3 The Called SS-user Reference PV field shall be as defined by the called SS-user.

8.3.4.4 The Common Reference PV field shall be as defined by the called SS-user.

8.3.4.5 The Additional Reference Information PV field shall be as defined by the called SS-user.

8.3.4.6 If the Connect/Accept Item is absent, the default values defined for the enclosed PI units shall apply.

8.3.4.7 The Protocol Options PV field shall indicate whether or not the responder is able to receive extended concatenated SPDUs (see § 6.3.7). The encoding and default for this field is defined in § 8.3.1.7.

8.3.4.8 The TSDU Maximum Size parameter shall be present if a TSDU maximum size is proposed by the receiver. The encoding and default for this field is defined in § 8.3.1.8. If an OVERFLOW ACCEPT SPDU has been sent previously, the TSDU maximum size parameter shall have the same value as was indicated on the OVERFLOW ACCEPT SPDU.

TABLE 14/X.225

Parameters of the ACCEPT SPDU

PGI	m/nm	Code	PI	m/nm	Code	Length	Reference
Connection identifier	nm	1	Called SS-user reference	nm	9	64 octets maximum	§ 7.4.1 a) 1) § 8.3.4.3
			Common reference	nm	11	64 octets maximum	§ 7.4.1 a) 2) § 8.3.4.4
			Additional reference information	nm	12	4 octets maximum	§ 7.4.1 a) 3) § 8.3.4.5
Connect/Accept item (see § 8.3.4.6)	nm	5	Protocol options	m	19	1 octet	§ 7.4.1 b) 1) § 8.3.4.7
			TSDU maximum size	nm	21	4 octets	§ 7.4.1 b) 2) § 8.3.4.8
			Version number	m	22	1 octet	§ 7.4.1 b) 3) § 8.3.4.9
			Initial serial number	nm	23	6 octets maximum	§ 7.4.1 b) 4) § 8.3.4.10
			Token setting item	nm	26	1 octet	§ 7.4.1 b) 5) § 8.3.4.11
			Token item	nm	16	1 octet	§ 7.4.1 c) § 8.3.4.12
			Session user requirements	nm	20	2 octets	§ 7.4.1 d) § 8.3.4.13
			Calling session selector	nm	51	16 octets maximum	§ 7.4.1 e) § 8.3.4.14
			Responding session selector	nm	52	16 octets maximum	§ 7.4.1 e) § 8.3.4.15
			Enclosure item	nm	25	1 octet	§ 7.37.1 § 8.3.4.17
User data	nm	193				See reference	§ 7.4.1 f) § 8.3.4.16

m: mandatory.

nm: not mandatory (see § 8.2.6).

8.3.4.9 The Version Number PV field shall have the value and encoding specified in § 8.3.1.9.

If an OVERFLOW ACCEPT SPDU has been sent previously on this session connection then the Version Number parameter shall have the same value as was indicated in the OVERFLOW ACCEPT SPDU.

8.3.4.10 The Initial Serial Number PV field shall only be present if the activity management functional unit is not selected and if any of the following functional units are selected:

a) minor synchronize functional unit;

b) major synchronize functional unit;

c) resynchronize functional unit.

The encoding for the Initial Serial Number PV field is defined in § 8.3.1.10.

8.3.4.11 The Token Setting Item PV field indicates the initial token settings for each token available on this session connection. The bits and encoding are defined in § 8.3.1.11. In the case where the initial assignment of the related token was indicated as the called SS-user's choice (in the Token Setting Item PV field of the associated CONNECT SPDU), the field shall contain the value chosen by the called SS-user. Otherwise, the values set in the CONNECT SPDU must be returned. The value "called SS-user's choice" is not a permitted value in the ACCEPT SPDU. The values are relevant only if the appropriate functional units are requested in the Session User Requirements parameter. If no functional unit requiring a token has been requested, this parameter need not be present.

8.3.4.12 The Token Item PV field, if present, shall indicate which tokens are requested by the called SS-user:

a) bit 7 = 1 release token;

b) bit 5 = 1 major/activity token;

c) bit 3 = 1 synchronize-minor token;

d) bit 1 = 1 data token.

Bits 2, 4, 6 and 8 are reserved.

Bits corresponding to tokens which are not available are ignored.

8.3.4.13 The bits in the Session User Requirements PV field shall indicate the functional units proposed by the called SS-user, for use over this session connection. This PV field shall not have both bit 1 set (half-duplex functional unit) and bit 2 set (duplex functional unit), but the chosen bit must have been set in the CONNECT SPDU. The encoding and default value is defined in § 8.3.1.12.

8.3.4.14 The Calling Session Selector, if present, shall be the same value as in the CONNECT SPDU.

8.3.4.15 The Responding Session Selector, if present, shall be derived from the Responding Session Address supplied by the responding SS-user. When this parameter is absent, the default shall be as though this parameter is set to a null value.

8.3.4.16 The length of the User Data parameter is limited such that the total length (including SI and LI) of the SPDU does not exceed 65 539 octets. If the Enclosure Item parameter is present, the User Data parameter is mandatory.

8.3.4.17 The Enclosure Item parameter, if present, shall indicate that the SPDU is the beginning, but not end of the SSDU. This parameter shall not be present if protocol version 1 is selected. The encoding for this field shall be:

a) bit 1 = 1 beginning of SSDU;

b) bit 2 = 0 not end of SSDU.

Bits 3-8 are reserved.

See § 8.4.2 for encoding subsequent SPDUs in the sequence.

8.3.5 *REFUSE (RF) SPDU*

8.3.5.1 The SI field shall contain the value 12.

8.3.5.2 The parameter fields shall be as specified in Table 15/X.225.

TABLE 15/X.225

Parameters of the REFUSE SPDU

PGI	m/nm	Code	PI	m/nm	Code	Length	Reference
Connection identifier	nm	1	Called SS-user reference	nm	9	64 octets maximum	§ 7.5.1 a) 1) § 8.3.5.3
			Common reference	nm	11	64 octets maximum	§ 7.5.1 a) 2) § 8.3.5.4
			Additional reference information	nm	12	4 octets maximum	§ 7.5.1 a) 3) § 8.3.5.5
			Transport disconnect	nm	17	1 octet	§ 7.5.1 b) § 8.3.5.6
			Session user requirements	nm	20	2 octets	§ 7.5.1 c) § 8.3.5.7
			Version number	nm	22	1 octet	§ 7.5.1 d) § 8.3.5.8
			Enclosure item	nm	25	1 octet	§ 7.37.1 § 8.3.5.10
			Reason code	nm	50	See reference	§ 7.5.1 e) § 8.3.5.9

m: mandatory.
nm: not mandatory (see § 8.2.6).

8.3.5.3 The Called SS-user Reference PV field shall be as defined by the called SS-user.

8.3.5.4 The Common Reference PV field shall be as defined by the called SS-user.

8.3.5.5 The Additional Reference Information PV field shall be as defined by the called SS-user.

8.3.5.6 The Transport Disconnect PV field shall indicate whether or not the transport connection is to be kept. The encoding for this field shall be:

a) bit 1 = 0 transport connection is kept;

b) bit 1 = 1 transport connection is released.

Bits 2-8 are reserved.

If this parameter is absent, the transport connection is released.

8.3.5.7 The Session User Requirements PV field shall only be present if the Reason Code is 2 and shall indicate the functional units required by the called SS-user and supported by the responder. The encoding shall be the same as in the CONNECT SPDU (see § 8.3.1.12).

8.3.5.8 The Version Number PV field shall have the value and encoding specified in § 8.3.1.9. If an OVERFLOW ACCEPT SPDU has been sent previously on this session connection, then the Version Number parameter shall have the same value as was indicated in the OVERFLOW ACCEPT SPDU.

8.3.5.9 The Reason Code PV field shall contain a reason code in the first octet. Depending on the value of this first octet, additional octets may be used. The following values are defined for the first octet:

a) 0 : rejection by the called SS user; reason not specified;

b) 1 : rejection by called SS-user due to temporary congestion;

c) 2 : rejection by called SS-user. The following octets may be used for user data up to a length such that the total length (including SI and LI) of the SPDU does not exceed 65 539 octets;

d) *128 + 1 : session selector unknown;

e) *128 + 2 : SS-user not attached to SSAP;

f) 128 + 3 : SPM congestion at connect time;

g) *128 + 4 : proposed protocol versions not supported;

h) 128 + 5 : rejection by the SPM; reason not specified;

i) 128 + 6 : rejection by the SPM; implementation restriction stated in the PICS.

Note – Reasons marked with an asterisk (*) may be reported to the SS-user as persistent, others reported as transient.

All other values are reserved.

The Session User Requirements parameter may only be present if the value of the reason code is 2. If the reason code has the value 2 and the Session User Requirements parameter is not present, the default value shall be assumed (see § 8.3.1.12).

If the Enclosure Item parameter is present, the Reason Code parameter is mandatory and shall be followed by octets of user data.

8.3.5.10 The Enclosure Item parameter, if present, shall be encoded as specified in § 8.3.4.17. This parameter shall not be present if protocol version 1 is selected.

8.3.6 *FINISH (FN) SPDU*

8.3.6.1 The SI field shall contain the value 9.

8.3.6.2 The parameter fields shall be as specified in Table 16/X.225.

TABLE 16/X.225

Parameters of the FINISH SPDU

PGI	m/nm	Code	PI	m/nm	Code	Length	Reference
			Transport disconnect	nm	17	1 octet	§ 7.6.1 a) § 8.3.6.3
			Enclosure item	nm	25	1 octet	§ 7.37.1 § 8.3.6.5
User data	nm	193				See reference	§ 7.6.1 b) § 8.3.6.4

m: mandatory.

nm: not mandatory (see § 8.2.6).

8.3.6.3 The Transport Disconnect PV field shall indicate whether or not the transport connection is to be kept. The encoding for this field shall be:

a) bit 1 = 0 transport connection is kept;

b) bit 1 = 1 transport connection is released.
Bits 2-8 are reserved.

If this parameter is absent, the transport connection shall be released.

8.3.6.4 The User Data PV field, if present, shall contain user data supplied by the SS-user.

The length of the User Data parameter is limited such that the total length (including SI and LI) of the SPDU does not exceed 65 539 octets.

8.3.6.5 The Enclosure Item parameter, if present, shall be encoded as specified in § 8.3.4.17. This parameter shall not be present if protocol version 1 is selected.

8.3.7 *DISCONNECT (DN) SPDU*

8.3.7.1 The SI field shall contain the value 10.

8.3.7.2 The parameter field shall be as specified in Table 17/X.225.

TABLE 17/X.225

Parameters of the DISCONNECT SPDU

PGI	m/nm	Code	PI	m/nm	Code	Length	Reference
			Enclosure item	nm	25	1 octet	§ 7.37.1 § 8.3.7.4
User data	nm	193				See reference	§ 7.7.1 § 8.3.7.3

m: mandatory.
nm: not mandatory (see § 8.2.6).

8.3.7.3 The User Data PV field, if present, shall contain user data supplied by the SS-user.

The length of the User Data parameter is limited such that the total length (including SI and LI) of the SPDU does not exceed 65 539 octets.

8.3.7.4 The Enclosure Item parameter, if present, shall be encoded as specified in § 8.3.4.17. This parameter shall not be present if protocol version 1 is selected.

8.3.8 *NOT FINISHED (NF) SPDU*

8.3.8.1 The SI field shall contain the value 8.

8.3.8.2 The parameter field shall be as specified in Table 18/X.225.

TABLE 18/X.225

Parameters of the NOT FINISHED SPDU

PGI	m/nm	Code	PI	m/nm	Code	Length	Reference
			Enclosure item	nm	25	1 octet	§ 7.37.1 § 8.3.8.4
User data	nm	193				See reference	§ 7.8.1 § 8.3.8.3

m: mandatory.
nm: not mandatory (see § 8.2.6).

8.3.8.3 The User Data PV field, if present, shall contain user data supplied by the SS-user. The length of the User Data parameter is limited such that the total length (including SI and LI) of the SPDU does not exceed 65 539 octets.

8.3.8.4 The Enclosure Item parameter, if present, shall be encoded as specified in § 8.3.4.17. This parameter shall not be present if protocol version 1 is selected.

8.3.9 *ABORT (AB) SPDU*

8.3.9.1 The SI field shall contain the value 25.

8.3.9.2 The parameter fields shall be as specified in Table 19/X.225.

TABLE 19/X.225

Parameters of the ABORT SPDU

PGI	m/nm	Code	PI	m/nm	Code	Length	Reference
			Transport disconnect	m	17	1 octet	§ 7.9.1 a) § 8.3.9.3
			Reflect parameter values	nm	49	9 octets maximum	§ 7.9.1 b) § 8.3.9.4
			Enclosure item	nm	25	1 octet	§ 7.37.1 § 8.3.9.6
User data	nm	193				See reference	§ 7.9.1 c) § 8.3.9.5

m: mandatory.
nm: not mandatory (see § 8.2.6).

8.3.9.3 The Transport Disconnect PV field shall indicate whether or not the transport connection is to be kept, together with an optional reason code. The encoding for this field shall be:

a) bit 1 = 0 transport connection is kept;

b) bit 1 = 1 transport connection is released;

c) bit 2 = 1 user abort (see § 8.3.9.5);

d) bit 3 = 1 protocol error (see § 8.3.9.4);

e) bit 4 = 1 no reason;

f) bit 5 = 1 implementation restriction stated in the PICS.

Bits 6-8 are reserved.

8.3.9.4 The Reflect Parameter Values PV field shall only be present if the Transport Disconnect PV field indicates protocol error and shall contain an implementation defined value and semantics.

8.3.9.5 The User Data PV field shall only be present if the Transport Disconnect PV field indicates user abort and shall contain user data supplied by the SS-user.

If this SPDU is to be sent on the transport expedited flow, the length of the User Data parameter is limited to 9 octets and the Enclosure Item shall not be present. If the SPDU is to be sent on the transport normal flow, the length of the User Data parameter is limited such that the total length (including SI and LI) of the SPDU does not exceed 65 539 octets.

8.3.9.6 The Enclosure Item parameter, if present, shall be encoded as specified in § 8.3.4.17. This parameter shall not be present if protocol version 1 is selected.

8.3.10 *ABORT ACCEPT (AA) SPDU*

8.3.10.1 The SI field shall contain the value 26.

8.3.10.2 There is no parameter field associated with this SPDU.

8.3.11 *DATA TRANSFER (DT) SPDU*

8.3.11.1 The SI field shall contain the value 1.

8.3.11.2 The parameter field shall be as specified in Table 20/X.225.

TABLE 20/X.225

Parameters of the DATA TRANSFER SPDU

PGI	m/nm	Code	PI	m/nm	Code	Length	Reference
			Enclosure item	nm	25	1 octet	§ 7.11.1 a) § 8.3.10.3
User information field						unlimited	§ 7.11.1 b) § 8.3.10.4

m: mandatory.

nm: not mandatory (see § 8.2.6).

8.3.11.3 The Enclosure Item PV field, if present, shall indicate whether or not this SPDU is the beginning or end of the SSDU. This field shall be present if segmenting has been selected. This field shall not be present if segmenting has not been selected. The encoding for this field shall be:

a) bit 1 = 1 beginning of SSDU;

bit 1 = 0 not beginning of SSDU;

b) bit 2 = 1 end of SSDU;
bit 2 = 0 not end of SSDU.
Bits 3-8 are reserved.

If this field is not present, the default shall be as though bit 1 = 1 and bit 2 = 1 (i.e. beginning and end of SSDU).

8.3.11.4 The User Information Field, if present, shall contain user data supplied by the SS-user. The User Information Field shall be present if the Enclosure Item is not present, or has bit 2 = 0.

8.3.12 *EXPEDITED (EX) SPDU*

8.3.12.1 The SI field shall contain the value 5.

8.3.12.2 This SPDU contains only a User Information Field as specified in Table 21/X.225.

TABLE 21/X.225

Parameters of the EXPEDITED SPDU

PGI	m/nm	Code	PI	m/nm	Code	Length	Reference
User information field						14 octets maximum	§ 7.12.1 § 8.3.12.3

m: mandatory.
nm: not mandatory (see § 8.2.6).

8.3.12.3 The User Information Field shall contain user data supplied by the SS-user.

8.3.13 *TYPED DATA (TD) SPDU*

8.3.13.1 The SI field shall contain the value 33.

8.3.13.2 The parameter field shall be as specified in Table 22/X.225.

TABLE 22/X.225

Parameters of the TYPED DATA SPDU

PGI	m/nm	Code	PI	m/nm	Code	Length	Reference
			Enclosure item	nm	25	1 octet	§ 7.13.1 a) § 8.3.13.3
User information field						unlimited	§ 7.13.1 b) § 8.3.13.4

m: mandatory.
nm: not mandatory (see § 8.2.6).

8.3.13.3 The Enclosure Item PV field, if present, shall indicate whether or not this SPDU is the beginning or end of the SSDU. This field shall be present if segmenting has been selected. This field shall not be present if segmenting has not been selected. The encoding for this field shall be:

a) bit 1 = 1 beginning of SSDU;
bit 1 = 0 not beginning of SSDU;

b) bit 2 = 1 end of SSDU;
bit 2 = 0 not end of SSDU.

Bits 3-8 are reserved.

If this field is not present, the default shall be as though bit 1 = 1 and bit 2 = 1 (i.e. beginning and end of SSDU).

8.3.13.4 The User Information Field, if present, shall contain user data supplied by the SS-user. The User Information Field shall be present if the Enclosure Item is not present, or has bit 2 = 0.

8.3.14 *CAPABILITY DATA (CD) SPDU*

8.3.14.1 The SI field shall contain the value 61.

8.3.14.2 The parameter field shall be as specified in Table 23/X.225.

TABLE 23/X.225

Parameters of the CAPABILITY DATA SPDU

PGI	m/nm	Code	PI	m/nm	Code	Length	Reference
			Enclosure item	nm	25	1 octet	§ 7.37.1 § 8.3.14.4
User data	nm	193				See reference	§ 7.14.1 § 8.3.14.3

m: mandatory.
nm: not mandatory (see § 8.2.6).

8.3.14.3 The User Data PV field, if present, shall contain user data supplied by the SS-user.

The length of the User Data parameter is limited such that the total length (including SI and LI) of the SPDU does not exceed 65 539 octets.

8.3.14.4 The Enclosure Item parameter, if present, shall be encoded as specified in § 8.3.4.17. This parameter shall not be present if protocol version 1 is selected.

8.3.15 *CAPABILITY DATA ACK (CDA) SPDU*

8.3.15.1 The SI field shall contain the value 62.

8.3.15.2 The parameter field shall be as specified in Table 24/X.225.

TABLE 24/X.225

Parameters of the CAPABILITY DATA ACK SPDU

PGI	m/nm	Code	PI	m/nm	Code	Length	Reference
			Enclosure item	nm	25	1 octet	§ 7.37.1 § 8.3.15.4
User data	nm	193				See reference	§ 7.15.1 § 8.3.15.3

m: mandatory.
nm: not mandatory (see § 8.2.6).

8.3.15.3 The User Data PV field, if present, shall contain user data supplied by the SS-user.

The length of the User Data parameter is limited such that the total length (including SI and LI) of the SPDU does not exceed 65 539 octets.

8.3.15.4 The Enclosure Item parameter, if present, shall be encoded as specified in § 8.3.4.17. This parameter shall not be present if protocol version 1 is selected.

8.3.16 *GIVE TOKENS (GT) SPDU*

8.3.16.1 The SI field shall contain the value 1.

8.3.16.2 The parameter field shall be as specified in Table 25/X.225.

TABLE 25/X.225

Parameters of the GIVE TOKENS SPDU

PGI	m/nm	Code	PI	m/nm	Code	Length	Reference
			Token item	nm	16	1 octet	§ 7.16.1 § 8.3.16.3
			Enclosure item	nm	25	1 octet	§ 7.37.1 § 8.3.16.5
User data	nm	193				See reference	§ 7.16.1 b)

m: mandatory.
nm: not mandatory (see § 8.2.6).

8.3.16.3 The Token Item PV field, if present, shall indicate which tokens are being given by the sending SS-user:

a) bit 7 = 1 release token;

b) bit 5 = 1 major/activity token;

c) bit 3 = 1 synchronize-minor token;

d) bit 1 = 1 data token.

Bits 2, 4, 6 and 8 are reserved.

Bits corresponding to tokens which are not available are ignored.

If this PV field is present, at least one bit corresponding to an available token shall be set to one.

8.3.16.4 This SPDU may be used without the Token Item PI unit when concatenated with Category 2 SPDUs according to Tables 7/X.225 and 8/X.225. With some concatenations (see Tables 7/X.225 and 8/X.225) the Token Item PI unit must be absent.

8.3.16.5 The Enclosure Item parameter, if present, shall be encoded as specified in § 8.3.4.17. This parameter shall not be present if protocol version 1 is selected.

8.3.16.6 The User Data PV field, if present, shall contain user data supplied by the SS-user. This PGI unit shall only be present if the Token Item PI unit is present and shall not be present if protocol version 1 is selected. The length of the User Data parameter is limited such that the total length (including SI and LI) does not exceed 65 539 octets.

8.3.17 *PLEASE TOKENS (PT) SPDU*

8.3.17.1 The SI field shall contain the value 2.

8.3.17.2 The parameter fields shall be as specified in Table 26/X.225.

TABLE 26/X.225

Parameters of the PLEASE TOKENS SPDU

PGI	m/nm	Code	PI	m/nm	Code	Length	Reference
			Token item	nm	16	1 octet	§ 7.17.1. a) § 8.3.17.3
			Enclosure item	nm	25	1 octet	§ 7.37.1 § 8.3.17.6
User data	nm	193				See reference	§ 7.17.1 b) § 8.3.17.4

m: mandatory.
nm: not mandatory (see § 8.2.6).

8.3.17.3 The Token Item PV field, if present, shall indicate which tokens are being requested by the sending SS-user:

a) bit 7 = 1 release token;

b) bit 5 = 1 major/activity token;

c) bit 3 = 1 synchronize-minor token;

d) bit 1 = 1 data token.

Bits 2, 4, 6 and 8 are reserved.

Bits corresponding to tokens which are not available are ignored.

If this PV field is present, at least one bit corresponding to an available token shall be set to one.

8.3.17.4 The User Data PV field, if present, shall contain user data supplied by the SS-user. This PGI unit shall only be present if the Token Item PI unit is present.

The length of the User Data parameter is limited such that the total length (including SI and LI) of the SPDU does not exceed 65 539 octets.

8.3.17.5 This SPDU may be used without the Token Item PI unit and the User Data PGI unit when concatenated with Category 2 SPDUs according to Tables 7/X.225 and 8/X.225. In this case, the SPDU does not achieve any Please Token function.

8.3.17.6 The Enclosure Item parameter, if present, shall be encoded as specified in § 8.3.4.17. This parameter shall not be present if protocol version 1 is selected.

8.3.18 *GIVE TOKENS CONFIRM (GTC) SPDU*

8.3.18.1 The SI field shall contain the value 21.

8.3.18.2 The parameter fields shall be as specified in Table 27/X.225.

TABLE 27/X.225

Parameters of the GIVE TOKENS CONFIRM SPDU

PGI	m/nm	Code	PI	m/nm	Code	Length	Reference
			Enclosure item	nm	25	1 octet	§ 7.37.1 § 8.3.18.3
User data	nm	193				See reference	§ 7.18.1 § 8.3.18.4

m: mandatory.
nm: not mandatory (see § 8.2.6).

8.3.18.3 The Enclosure Item parameter, if present, shall be encoded as specified in § 8.3.4.17. This parameter shall not be present if protocol version 1 is selected.

8.3.18.4 The User Data PV field, if present, shall contain user data supplied by the SS-user. This PGI unit shall not be present if protocol version 1 is selected. The length of the User Data parameter is limited such that the total length (including SI and LI) of the SPDU does not exceed 65 539 octets.

8.3.19 *GIVE TOKENS ACK (GTA) SPDU*

8.3.19.1 The SI field shall contain the value 22.

8.3.19.2 There is no parameter field associated with this SPDU.

8.3.20 *MINOR SYNC POINT (MIP) SPDU*

8.3.20.1 The SI field shall contain the value 49.

8.3.20.2 The parameter fields shall be as specified in Table 28/X.225.

TABLE 28/X.225

Parameters of the MINOR SYNC POINT SPDU

PGI	m/nm	Code	PI	m/nm	Code	Length	Reference
			Sync type item	nm	15	1 octet	§ 7.20.1 a) § 8.3.20.3
			Serial number	m	42	6 octets maximum	§ 7.20.1 b) § 8.3.20.4
			Enclosure item	nm	25	1 octet	§ 7.37.1 § 8.3.20.6
User data	nm	193				See reference	§ 7.20.1 c) § 8.3.20.5

m: mandatory.
nm: not mandatory (see § 8.2.6).

8.3.20.3 The Sync Type Item PV field, if present, shall indicate that an explicit confirmation is not required:

bit 1 = 1 explicit confirmation not required.

Bits 2-8 are reserved.

This parameter field shall be absent if an explicit confirmation is required.

8.3.20.4 The Serial Number PV field shall be coded as specified in § 8.3.1.10.

8.3.20.5 The User Data PV field, if present, shall contain user data supplied by the SS-user.

The length of the User Data parameter is limited such that the total length (including SI and LI) of the SPDU does not exceed 65 539 octets.

8.3.20.6 The Enclosure Item parameter, if present, shall be encoded as specified in § 8.3.4.17. This parameter shall not be present if protocol version 1 is selected.

8.3.21 *MINOR SYNC (MIA) SPDU*

8.3.21.1 The SI field shall contain the value 50.

8.3.21.2 The parameter fields shall be as specified in Table 29/X.225.

TABLE 29/X.225

Parameters of the MINOR SYNC ACK SPDU

PGI	m/nm	Code	PI	m/nm	Code	Length	Reference
			Serial number	m	42	6 octets maximum	§ 7.21.1 a) § 8.3.21.3
			Enclosure item	nm	25	1 octet	§ 7.37.1 § 8.3.21.5
			User data	nm	46	See reference	§ 7.21.1 b) § 8.3.21.4

m: mandatory.
nm: not mandatory (see § 8.2.6).

8.3.21.3 The Serial Number PV field shall be coded as specified in § 8.3.1.10.

8.3.21.4 The User Data PV field, if present, shall contain user data supplied by the SS-user.

The length of the User Data parameter is limited such that the total length (including SI and LI) of the SPDU does not exceed 65 539 octets.

8.3.21.5 The Enclosure Item parameter, if present, shall be encoded as specified in § 8.3.4.17. This parameter shall not be present if protocol version 1 is selected.

8.3.22 *MAJOR SYNC POINT (MAP) SPDU*

8.3.22.1 The SI field shall contain the value 41. This is the same value as the SI field for the ACTIVITY END SPDU (see § 8.3.35).

8.3.22.2 The parameter fields shall be as specified in Table 30/X.225.

TABLE 30/X.225

Parameters of the MAJOR SYNC POINT SPDU

PGI	m/nm	Code	PI	m/nm	Code	Length	Reference
			Sync type item	m	15	1 octet	§ 7.22.1 a) § 8.3.22.3
			Serial number	m	42	6 octets maximum	§ 7.22.1 b) § 8.3.22.4
			Enclosure item	nm	25	1 octet	§ 7.37.1 § 8.3.22.6
User data	nm	193				See reference	§ 7.22.1 c) § 8.3.22.5

m: mandatory.
nm: not mandatory (see § 8.2.6).

8.3.22.3 The Sync Type Item PV field shall indicate that this is not the end of an activity:

bit 1 = 1 major synchronization point without end of activity.

Bits 2-8 are reserved.

8.3.22.4 The Serial Number PV field shall be coded as specified in § 8.3.1.10.

8.3.22.5 The User Data PV field, if present, shall contain user data supplied by the SS-user.

The length of the User Data parameter is limited such that the total length (including SI and LI) of the SPDU does not exceed 65 539 octets.

8.3.22.6 The Enclosure Item parameter, if present, shall be encoded as specified in § 8.3.4.17. This parameter shall not be present if protocol version 1 is selected.

8.3.23 *MAJOR SYNC ACK (MAA) SPDU*

8.3.23.1 The SI field shall contain the value 42.

8.3.23.2 The parameter fields shall be as specified in Table 31/X.225.

TABLE 31/X.225

Parameters of the MAJOR SYNC ACK SPDU

PGI	m/nm	Code	PI	m/nm	Code	Length	Reference
			Serial number	m	42	6 octets maximum	§ 7.23.1 a) § 8.3.23.3
			Enclosure item	nm	25	1 octet	§ 7.37.1 § 8.3.23.5
User data	nm	193				See reference	§ 7.23.1 b) § 8.3.23.4

m: mandatory.
nm: not mandatory (see § 8.2.6).

8.3.23.3 The Serial Number PV field shall be coded as specified in § 8.3.1.10.

8.3.23.4 The User Data PV field, if present, shall contain user data supplied by the SS-user.

Note – This SPDU is identical to the ACTIVITY END ACK SPDU (see § 8.3.36).

The length of the User Data parameter is limited such that the total length (including SI and LI) of the SPDU does not exceed 65 539 octets.

8.3.23.5 The Enclosure Item parameter, if present, shall be encoded as specified in § 8.3.4.17. This parameter shall not be present if protocol version 1 is selected.

8.3.24 *RESYNCHRONIZE (RS) SPDU*

8.3.24.1 The SI field shall contain the value 53.

8.3.24.2 The parameter fields shall be as specified in Table 32/X.245.

8.3.24.3 The Token Setting Item PV field indicates the requesting SS-user's proposed settings for each available token. The bits of the Token Setting Item PV field are defined as bit pairs:

a) bits 8, 7 release token;

b) bits 6, 5 major/activity token;

c) bits 4, 3 synchronize-minor token;

d) bits 2, 1 data token.

The encoding for each bit pair shall be:

e) 00 : requestor's side;

f) 01 : acceptor's side;

g) 10 : accepting SS-user's choice;

h) 11 : reserved.

The values are relevant only if the appropriate token is available. If no token is available, this parameter need not be present.

TABLE 32/X.225

Parameters of the RESYNCHRONIZE SPDU

PGI	m/nm	Code	PI	m/nm	Code	Length	Reference
			Token setting item	nm	26	1 octet	§ 7.24.1 a) § 8.3.24.3
			Resync type	m	27	1 octet	§ 7.24.1 b) § 8.3.24.4
			Serial number	m	42	6 octets maximum	§ 7.24.1 c) § 8.3.24.5
			Enclosure item	nm	25	1 octet	§ 7.37.1 § 8.3.24.7
User data	nm	193				See reference	§ 7.24.1 d) § 8.3.24.6

m: mandatory.
nm: not mandatory (see § 8.2.6).

8.3.24.4 The Resync Type PV field indicates the resynchronize type which is required:

a) 0 : resynchronize restart;

b) 1 : resynchronize abandon;

c) 2 : resynchronize set.

All other values are reserved.

8.3.24.5 The Serial Number PV field shall be encoded as specified in § 8.3.1.10.

8.3.24.6 The User Data PV field, if present, shall contain user data supplied by the SS-user.

The length of the User Data parameter is limited such that the total length (including SI and LI) of the SPDU does not exceed 65 539 octets.

8.3.24.7 The Enclosure Item parameter, if present, shall be encoded as specified in § 8.3.4.17. This parameter shall not be present if protocol version 1 is selected.

8.3.25 *RESYNCHRONIZE ACK (RA) SPDU*

8.3.25.1 The SI field shall contain the value 34.

8.3.25.2 The parameter fields shall be as specified in Table 33/X.245.

TABLE 33/X.225

Parameters of the RESYNCHRONIZE ACK SPDU

PGI	m/nm	Code	PI	m/nm	Code	Length	Reference
			Token setting item	nm	26	1 octet	§ 7.25.1 a) § 8.3.25.3
			Serial number	m	42	6 octets maximum	§ 7.25.1 b) § 8.3.25.4
			Enclosure item	nm	25	1 octet	§ 7.37.1 § 8.3.25.6
User data	nm	193				See reference	§ 7.25.1 c) § 8.3.25.5

m: mandatory.
nm: not mandatory (see § 8.2.6).

8.3.25.3 The Token Setting Item PV field indicates token settings for each token available on the session connection. The bits and encoding are defined in § 8.3.24.3. For the case where the requesting SS-user has indicated that the assignment is the accepting SS-user's choice, the field shall contain the values chosen by the accepting SS-user. Otherwise, the values in the RESYNCHRONIZE SPDU must be returned.

This parameter need not be present if no tokens are available.

8.3.25.4 The Serial Number PV field shall be coded as specified in § 8.3.1.10.

8.3.25.5 The User Data PV field, if present, shall contain user data supplied by the SS-user.

The length of the User Data parameter is limited such that the total length (including SI and LI) of the SPDU does not exceed 65 539 octets.

8.3.25.6 The Enclosure Item parameter, if present, shall be encoded as specified in § 8.3.4.17. This parameter shall not be present if protocol version 1 is selected.

8.3.26 *PREPARE (PR) SPDU*

8.3.26.1 The SI field shall contain the value 7.

8.3.26.2 The parameter field shall be as specified in Table 34/X.225.

TABLE 34/X.225

Parameters of the PREPARE SPDU

PGI	m/nm	Code	PI	m/nm	Code	Length	Reference
			Prepare type	m	24	1 octet	§ 7.26.1 § 8.3.26.3

m: mandatory.
nm: not mandatory (see § 8.2.6).

8.3.26.3 The Prepare Type PV field indicates which SPDU should be expected on the transport normal flow. The value for this field shall be:

a) 1 Prepare for MAJOR SYNC ACK SPDU;

b) 2 Prepare for RESYNCHRONIZE SPDU;

c) 3 Prepare for RESYNCHRONIZE ACK SPDU;

d) 4 Prepare for ABORT SPDU.

All other values are reserved and shall not be used.

8.3.27 *EXCEPTION REPORT (ER) SPDU*

8.3.27.1 The SI field shall contain the value 0.

8.3.27.2 The parameter field shall be as specified in Table 35/X.225.

TABLE 35/X.225

Parameters of the EXCEPTION REPORT SPDU

PGI	m/nm	Code	PI	m/nm	Code	Length	Reference
			Reflect parameter values	m	49	65 531 octets maximum	§ 7.27.1 § 8.3.27.3

m: mandatory.
nm: not mandatory (see § 8.2.6).

8.3.27.3 The Reflect Parameter Values PV field shall contain the bit pattern of the SPDU in error, up to and including the detected error, to a maximum of *n* octets,

$$\text{where } 1024 \leqslant n \leqslant 65\,531$$

Note – Not all implementations may be able to deal with field lengths greater than 1024. It is recommended that, whenever possible, the Reflect Parameters PV field should contain the bit pattern of the SPDU in error up to and including the detected error.

8.3.28 *EXCEPTION DATA (ED) SPDU*

8.3.28.1 The SI field shall contain the value 48.

8.3.28.2 The parameter fields shall be as specified in Table 36/X.225.

8.3.28.3 The Reason Code PV field shall contain one of the following values:

a) 0 No specific reason stated;

b) 1 Temporarily unable to continue;

c) 2 Reserved;

d) 3 User sequence error;

e) 4 Reserved;

f) 5 Local SS-user error;

g) 6 Unrecoverable procedural error;

h) 128 Demand data token.

All other values are reserved and shall not be used.

TABLE 36/X.225

Parameters of the EXCEPTION DATA SPDU

PGI	m/nm	Code	PI	m/nm	Code	Length	Reference
			Reason code	m	50	1 octet	§ 7.28.1 a) § 8.3.25.3
			Enclosure item	nm	25	1 octet	§ 7.37.1 § 8.3.28.5
User data	nm	193				See reference	§ 7.28.1 b) § 8.3.28.4

m: mandatory.
nm: not mandatory (see § 8.2.6).

8.3.28.4 The User Data PV field, if present, shall contain user data supplied by the SS-user.

The length of the User Data parameter is limited such that the total length (including SI and LI) of the SPDU does not exceed 65 539 octets.

8.3.28.5 The Enclosure Item parameter, if present, shall be encoded as specified in § 8.3.4.17. This parameter shall not be present if protocol version 1 is selected.

8.3.29 *ACTIVITY START (AS) SPDU*

8.3.29.1 The SI field shall contain the value 45.

8.3.29.2 The parameter fields shall be as specified in Table 37/X.225.

TABLE 37/X.225

Parameters of the ACTIVITY START SPDU

PGI	m/nm	Code	PI	m/nm	Code	Length	Reference
			Activity identifier	m	41	6 octets maximum	§ 7.29.1 a) § 8.3.29.3
			Enclosure item	nm	25	1 octet	§ 7.37.1 § 8.3.29.5
User data	nm	193				See reference	§ 7.29.1 b) § 8.3.29.4

m: mandatory.
nm: not mandatory (see § 8.2.6).

8.3.29.3 The Activity Identifier PV field shall be as defined by the sending SS-user.

8.3.29.4 The User Data PV field, if present, shall contain user data supplied by the SS-user.

The length of the User Data parameter is limited such that the total length (including SI and LI) of the SPDU does not exceed 65 539 octets.

8.3.29.5 The Enclosure Item parameter, if present, shall be encoded as specified in § 8.3.4.17. This parameter shall not be present if protocol version 1 is selected.

8.3.30 *ACTIVITY RESUME (AR) SPDU*

8.3.30.1 The SI field shall contain the value 29.

8.3.30.2 The parameter fields shall be as specified in Table 38/X.225.

8.3.30.3 The Called SS-user Reference PV field shall be as defined by the sending SS-user.

8.3.30.4 The Calling SS-user Reference PV field shall be as defined by the sending SS-user.

8.3.30.5 The Common Reference PV field shall be as defined by the sending SS-user.

8.3.30.6 The Additional Reference Information PV field shall be as defined by the sending SS-user.

8.3.30.7 The Old Activity Identifier PV field shall be as defined by the sending SS-user.

TABLE 38/X.225

Parameters of the ACTIVITY RESUME SPDU

PGI	m/nm	Code	PI	m/nm	Code	Length	Reference
Linking information	m	33	Called SS-user reference	nm	9	64 octets maximum	§ 7.30.1 a) 1) § 8.3.30.3
			Calling SS-user reference	nm	10	64 octets maximum	§ 7.30.1 a) 2) § 8.3.30.4
			Common reference	nm	11	64 octets maximum	§ 7.30.1 a) 3) § 8.3.30.5
			Additional reference information	nm	12	4 octets maximum	§ 7.30.1 a) 4) § 8.3.30.6
			Old activity identifier	m	41	6 octets maximum	§ 7.30.1 a) 5) § 8.3.30.7
			Serial number	m	42	6 octets maximum	§ 7.30.1 a) 6) § 8.3.30.8
			New activity identifier	m	41	6 octets maximum	§ 7.30.1 b) § 8.3.30.9
			Enclosure item	nm	25	1 octet	§ 7.37.1 § 8.3.30.11
User data	nm	193				See reference	§ 7.30.1 c) § 8.3.30.10

m: mandatory.

nm: not mandatory (see § 8.2.6).

8.3.30.8 The Serial Number PV field shall be coded as specified in § 8.3.1.10.

8.3.30.9 The New Activity Identifier PV field shall be as defined by the sending SS-user.

8.3.30.10 The User Data PV field, if present, shall contain user data supplied by the SS-user.

The length of the User Data parameter is limited such that the total length (including SI and LI) of the SPDU does not exceed 65 539 octets.

8.3.30.11 The Enclosure Item parameter, if present, shall be encoded as specified in § 8.3.4.17. This parameter shall not be present if protocol version 1 is selected.

8.3.31 *ACTIVITY INTERRUPT (AI) SDPU*

8.3.31.1 The SI field shall contain the value 25.

8.3.31.2 The parameter fields shall be as specified in Table 39/X.225.

TABLE 39/X.225

Parameters of the ACTIVITY INTERRUPT SPDU

PGI	m/nm	Code	PI	m/nm	Code	Length	Reference
			Reason code	nm	50	1 octet	§ 7.31.1 § 8.3.31.3
			Enclosure item	nm	25	1 octet	§ 7.37.1 § 8.3.31.4
User data	nm	193				See reference	§ 7.31.1 b) § 8.3.31.5

m: mandatory.
nm: not mandatory (see § 8.2.6).

8.3.31.3 The Reason Code PV field shall contain one of the following values:

a) 0 No specific reason stated;

b) 1 Temporarily unable to continue;

c) 2 Reserved;

d) 3 User sequence error;

e) 4 Reserved;

f) 5 Local SS-user error;

g) 6 Unrecoverable procedural error;

h) 128 Demand data token.

All other values are reserved and shall not be used.

8.3.31.4 The Enclosure Item parameter, if present, shall be encoded as specified in § 8.3.4.17. This parameter shall not be present if protocol version 1 is selected.

8.3.31.5 The User Data PV field, if present, shall contain user data supplied by the SS-user. This PGI unit shall not be present if protocol version 1 is selected.

The length of the User Data parameter is limited such that the total length (including SI and LI) of the SPDU does not exceed 65 539 octets.

8.3.32 *ACTIVITY INTERRUPT ACK (AIA) SPDU*

8.3.32.1 The SI field shall contain the value 26.

8.3.32.2 The parameter fields shall be as specified in Table 40/X.225.

TABLE 40/X.225

Parameters of the ACTIVITY INTERRUPT ACK SPDU

PGI	m/nm	Code	PI	m/nm	Code	Length	Reference
			Enclosure item	nm	25	1 octet	§ 7.37.1 § 8.3.32.3
User data	nm	193				See reference	§ 7.32.1 § 8.3.32.4

m: mandatory.
nm: not mandatory (see § 8.2.6).

8.3.32.3 The Enclosure Item parameter, if present, shall be encoded as specified in § 8.3.4.17. This parameter shall not be present if protocol version 1 is selected.

8.3.32.4 The User Data PV field, if present, shall contain user data supplied by the SS-user. This PGI unit shall not be present if protocol version 1 is selected.

The length of the User Data parameter is limited such that the total length (including SI and LI) of the SPDU does not exceed 65 539 octets.

8.3.33 *ACTIVITY DISCARD (AD) SPDU*

8.3.33.1 The SI field shall contain the value 57.

8.3.33.2 The parameter fields shall be as specified in Table 41/X.225.

TABLE 41/X.225

Parameters of the ACTIVITY DISCARD SPDU

PGI	m/nm	Code	PI	m/nm	Code	Length	Reference
			Reason code	nm	50	1 octet	§ 7.33.1 § 8.3.33.3
			Enclosure item	nm	25	1 octet	§ 7.37.1 § 8.3.33.4
User data	nm	193				See reference	§ 7.33.1 b) § 8.3.33.5

m: mandatory.
nm: not mandatory (see § 8.2.6).

8.3.33.3 The Reason Code PV field shall contain one of the following values:

a) 0 No specific reason stated;

b) 1 Temporarily unable to continue;

c) 2 Reserved;

d) 3 User sequence error;

e) 4 Reserved;

f) 5 Local SS-user error;

g) 6 Unrecoverable procedural error;

h) 128 Demand data token.

All other values are reserved and shall not be used.

8.3.33.4 The Enclosure Item parameter, if present, shall be encoded as specified in § 8.3.4.17. This parameter shall not be present if protocol version 1 is selected.

8.3.33.5 The User Data PV field, if present, shall contain user data supplied by the SS-user. This PGI unit shall not be present if protocol version 1 is selected.

The length of the User Data parameter is limited such that the total length (including SI and LI) of the SPDU does not exceed 65 539 octets.

8.3.34 *ACTIVITY DISCARD ACK (ADA) SPDU*

8.3.34.1 The SI field shall contain the value 58.

8.3.34.2 The parameter fields shall be as specified in Table 42/X.225.

TABLE 42/X.225

Parameters of the ACTIVITY DISCARD ACK SPDU

PGI	m/nm	Code	PI	m/nm	Code	Length	Reference
			Enclosure item	nm	25	1 octet	§ 7.37.1 - § 8.3.34.3
User data	nm	193				See reference	§ 7.34.1 § 8.3.34.4

m: mandatory.
nm: not mandatory (see § 8.2.6).

8.3.34.3 The Enclosure Item parameter, if present, shall be encoded as specified in § 8.3.4.17. This parameter shall not be present if protocol version 1 is selected.

8.3.34.4 The User Data PV field, if present, shall contain user data supplied by the SS-user. This PGI unit shall not be present if protocol version 1 is selected.

The length of the User Data parameter is limited such that the total length (including SI and LI) of the SPDU does not exceed 65 539 octets.

8.3.35 *ACTIVITY END (AE) SPDU*

8.3.35.1 The SI field shall contain the value 41. This is the same value as the SI field for the MAJOR SYNC POINT SPDU (see § 8.3.22).

8.3.35.2 The parameter fields shall be as specified in Table 43/X.225.

TABLE 43/X.225

Parameters of the ACTIVITY END SPDU

PGI	m/nm	Code	PI	m/nm	Code	Length	Reference
			Serial number	m	42	6 octets maximum	§ 7.36.1 a) § 8.3.35.3
			Enclosure item	nm	25	1 octet	§ 7.37.1 § 8.3.35.5
User data	nm	193				See reference	§ 7.36.1 b) § 8.3.35.4

m: mandatory.
nm: not mandatory (see § 8.2.6).

8.3.35.3 The Serial Number PV field shall be coded as specified in § 8.3.1.10.

8.3.35.4 The User Data PV field, if present, shall contain user data supplied by the SS-user.

The length of the User Data parameter is limited such that the total length (including SI and LI) of the SPDU does not exceed 65 539 octets.

8.3.35.5 The Enclosure Item parameter, if present, shall be encoded as specified in § 8.3.4.17. This parameter shall not be present if protocol version 1 is selected.

8.3.36 *ACTIVITY END ACK (AEA) SPDU*

The ACTIVITY END ACK SPDU is identical to the MAJOR SYNC ACK SPDU (see § 8.3.23).

8.4 *Additional encoding rules for segmented SSDUs*

ACCEPT SPDU

REFUSE SPDU

FINISH SPDU

DISCONNECT SPDU

NOT FINISHED SPDU

ABORT SPDU

CAPABILITY DATA SPDU

CAPABILITY DATA ACK SPDU

GIVE TOKENS SPDU

PLEASE TOKENS SPDU

GIVE TOKENS CONFIRM SPDU

MINOR SYNC POINT SPDU

MINOR SYNC POINT ACK SPDU

MAJOR SYNC POINT SPDU

MAJOR SYNC POINT ACK SPDU

RESYNCHRONIZE SPDU

RESYNCHRONIZE ACK SPDU

EXCEPTION DATA SPDU

ACTIVITY START SPDU

ACTIVITY RESUME SPDU

ACTIVITY INTERRUPT SPDU

ACTIVITY INTERRUPT ACK SPDU

ACTIVITY DISCARD SPDU

ACTIVITY DISCARD ACK SPDU

ACTIVITY END SPDU

ACTIVITY END ACK SPDU

8.4.1 *First SPDU in sequence*

The first SPDU in the sequence shall be as specified in § 8.3.

8.4.2 *Subsequent SPDUs in sequence*

8.4.2.1 For all SPDUs except the REFUSE SPDU and the MINOR SYNC ACK SPDU, the encoding shall be as follows:

8.4.2.1.1 The SI field shall have the same value as the SI field value of the initial SPDU of the sequence.

8.4.2.1.2 The parameter fields shall be as specified in Table 44/X.225.

8.4.2.1.3 The Enclosure Item PV field shall indicate whether or not this SPDU is the end of the SSDU. The encoding shall be:

a) bit 1 = 0 not beginning of SSDU;

b) bit 2 = 1 end of SSDU;
bit 2 = 0 not end of SSDU.
Bits 3-8 are reserved.

8.4.2.1.4 The User Data field, if present, shall contain a segment of the associated SSDU. The User Data field shall be present if the Enclosure Item has bit 2 = 0.

TABLE 44/X.225

Parameters of subsequent SPDUs when segmenting is required

PGI	m/nm	Code	PI	m/nm	Code	Length	Reference
			Enclosure item	m	25	1 octet	§ 7.37.1 § 8.4.2.1.3
User data	nm	193				65528 octets maximum	§ 7.37.1 § 8.4.2.1.4

m: mandatory.
nm: not mandatory (see § 8.2.6).

8.4.2.2 For the REFUSE SPDU the encoding shall be as follows:

8.4.2.2.1 The SI field shall have the same value as the SI field value of the initial SPDU of the sequence.

8.4.2.2.2 The parameter fields shall be as specified in Table 45/X.225.

8.4.2.2.3 The Enclosure Item PV field shall indicate whether or not this SPDU is the end of the SSDU. The encoding shall be:

a) bit 1 = 0 not beginning of SSDU;

b) bit 2 = 1 end of SSDU;
bit 2 = 0 not end of SSDU.
Bits 3-8 are reserved.

8.4.2.2.4 The Reason Code field, if present, shall contain a segment of the associated SSDU. The Reason Code field shall be present if the Enclosure Item has bit 2 = 0.

TABLE 45/X.225

Parameters of subsequent REFUSE SPDUs when segmenting is required

PGI	m/nm	Code	PI	m/nm	Code	Length	Reference
			Enclosure item	m	25	1 octet	§ 7.37.1 § 8.4.2.2.3
Reason code	nm				50	65 528 octets maximum	§ 7.37.1 § 8.4.2.2.4

m: mandatory.
nm: not mandatory (see § 8.2.6).

8.4.2.3 For the MINOR SYNC ACK SPDU the encoding shall be as follows:

8.4.2.3.1 The SI field shall have the same value as the SI field value of the initial SPDU of the sequence.

8.4.2.3.2 The parameter fields shall be as specified in Table 46/X.225.

8.4.2.3.3 The Enclosure Item PV field shall indicate whether or not this SPDU is the end of the SSDU. The encoding shall be:

a) bit 1 = 0 not beginning of SSDU;

b) bit 2 = 1 end of SSDU;
bit 2 = 0 not end of SSDU.
Bits 3-8 are reserved.

8.4.2.3.4 The User Data field, if present, shall contain a segment of the associated SSDU. The User Data field shall be present if the Enclosure Item has bit 2 = 0.

TABLE 46/X.225

Parameters of subsequent MINOR SYNC ACK SPDUs when segmenting is required

PGI	m/nm	Code	PI	m/nm	Code	Length	Reference
			Enclosure item	m	25	1 octet	§ 7.37.1 § 8.4.2.3.3
			User data	nm	46	65 528 octets maximum	§ 7.37.1 § 8.4.2.3.4

m: mandatory.
nm: not mandatory (see § 8.2.6).

9 Conformance to this standard

9.1 A system claiming conformance to this Recommendation shall exhibit external behaviour consistent with having implemented an SPM for the Kernel functional unit together with either or both of the Half-duplex and the Duplex functional units.

9.2 The system may exhibit external behaviour consistent with containing an implementation of any other functional unit provided that:

a) if the Capability Data functional unit is implemented, the Activity Management functional unit shall also be implemented; and

b) if the Exceptions functional unit is implemented, the Half-duplex functional unit shall also be implemented.

9.3 For all protocol versions which are claimed to be implemented, the system shall be capable of:

a) initiating a session connection (by sending a CONNECT SPDU) or responding to a CONNECT SPDU (according to the procedures specified in section 7), or both;

b) following all the remaining procedures in the Kernel functional unit; and

c) following all the procedures for each functional unit that the system claims to implement,

where following the procedures specified in b) and c) shall mean:

d) accepting all correct sequences of SPDUs received from peer equipment, and responding with correct SPDU sequences, for the defined states of a session connection;

e) responding correctly to all incorrect sequences of SPDUs received for a defined state of a session connection.

9.4 Claims of conformance shall state:

a) which functional units are implemented;

b) whether or not extended concatenation is implemented;

c) whether or not segmenting is implemented and, if segmenting is implemented, the maximum size of TSDU which the system is capable of handling;

d) whether or not the use of transport expedited service is implemented;

e) which protocol versions are implemented.

9.5 The implementor shall provide a Protocol Implementation Conformance Statement (PICS).

Note – In particular any limit imposed by an implementation on the number of octets of SS-user data which can be passed in a single session primitive shall be stated in the PICS.

ANNEX A

(to Recommendation X.225)

State tables

A.1 *General*

This Annex describes the session protocol in terms of state tables. The state tables show the state of a session connection, the events that occur in the protocol, the actions taken and the result state.

These state tables do not constitute a formal definition of the session protocol; they are included to provide a more precise specification of the elements of procedure described in § 7.

Table A-1/X.225 specifies the abbreviated name, category and name of each incoming event. The categories are SS-user event, TS-provider event, timer event and valid SPDU event.

Table A-2/X.225 specifies the abbreviated name and name of each state.

Table A-3/X.225 specifies the abbreviated name, category and name of each outgoing event. The categories are SS-provider event, TS-user event and SPDU event.

Table A-4/X.225 summarizes the operations on the variables (VA), V(M), V(R) and Vsc.

Table A-5/X.225 specifies the specific actions.

Table A-6/X.225 specifies the predicates.

Tables A-7/X.225 to A-15/X.225 specify the state tables.

A.2 *Notation for state tables*

A.2.1 Incoming events, states and outgoing events are represented by their abbreviated names.

A.2.2 Specific actions are represented by the notation [*n*], where *n* is the number of the specific action in Table A-5/X.225.

A.2.3 Notes are represented by the notation (*n*), where *n* is the number of the note at the foot of Table A-6/X.225.

A.2.4 Predicates are represented by the notation p*n*, where *n* is the number of the predicate in Table A-6/X.225.

A.2.5 Boolean operators are represented by the following notation:

&	AND
^	NOT
OR	OR

A.3 *Conventions for entries in state tables*

A.3.1 The intersection of each state and incoming event which is invalid is left blank.

A.3.2 The intersection of each state and incoming event which is valid contains entries which are either:

a) an *action list* which:

1) may contain outgoing events and/or specific actions;

2) always contains the resultant state;

or

b) one or more *conditional action lists*, each consisting of:

1) a predicate expression comprising predicates and Boolean operators;

2) an action list (as in § A.3.2 a)).

Note – The action lists and conditional action lists use the notation in § A.2.

A.3.3 The intersection of each state and incoming event which is not logically possible for the SPM is indicated by // in the top lefthand corner of the intersection.

Note – Such entries are a consequence of the tabular presentation of the state tables.

A.4 *Actions to be taken by the SPM*

The state tables define the action to be taken by the SPM.

A.4.1 *Invalid intersections*

If the intersection of the state and an incoming event is invalid, one of the following actions shall be taken.

A.4.1.1 If the incoming event comes from the SS-user, any action taken by the SPM is a local matter.

A.4.1.2 If the incoming event is related to a received SPDU and if the state of the transport connection makes it possible, the SPM shall either:

a) take the following actions:

1) issue an S-P-ABORT indication;

2) send an ABORT SPDU;

3) start the timer, TIM;

4) wait for a T-DISCONNECT indication or an ABORT ACCEPT SPDU (STA 16); or

b) if the following conditions hold:

1) the data token is available but not assigned to the SPM; and

2) – the activity management functional unit has not been selected; or

– the activity management functional unit has been selected and an activity is in progress; or

– the activity management functional unit has been selected and the SPM is in STA 22; and

3) the exceptions functional unit has been selected; and

4) the session connection is in the data transfer phase (i.e. states 4A, 4B, 5A, 5B, 5C, 6, 10A, 10B, 11A, 11B, 11C, 15A, 15B, 15C, 19, 20, 22, 713);

take the following actions:

5) send an EXCEPTION REPORT SPDU;

6) issue an S-P-EXCEPTION-REPORT indication;

7) enter STA 20 and wait for a recovery request or SPDU.

Note: – It should be noted that sending an EXCEPTION REPORT SPDU may lead to an SPM deadlock. It is therefore advised to send the ABORT SPDU rather than the EXCEPTION REPORT SPDU, especially in the case of protocol errors.

A.4.1.3 If the incoming event falls into neither of the above categories (including those which are impossible by the definition of the behaviour of the SPM or TS-provider) any action taken by the SPM is a local matter.

A.4.2 *Valid intersections*

If the intersection of the state and incoming event is valid, one of the following actions shall be taken.

A.4.2.1 If the intersection contains an action list, the SPM shall take the specific actions in the order specified in the state table.

A.4.2.2 If the intersection contains one or more conditional action lists, for each predicate expression that is true the SPM shall take the specific actions in the order given in the action list associated with the predicate expression. If none of the predicate expressions are true, the SPM shall take one of the actions defined in § A.4.1.

A.4.2.3 *Procedures for segmented SSDUs*

The state tables do not take account of segmented SSDUs. When an outgoing SSDU is to be segmented or an incoming SSDU is segmented, the procedures defined in § 7.37 apply to the outgoing event at the appropriate intersection of the state tables (that part of the action which transmits the SPDU).

A.4.3 *Receipt of SPDUs*

A.4.3.1 *Valid SPDUs*

The SPM shall process valid SPDUs as specified in Tables A-7/X.225 to A-15/X.225.

A.4.3.1.1 *Rules of extensibility*

This Recommendation does not specify the action to be taken in response to a PGI unit containing a PGI code listed in Annex C, or to a PI unit containing a PI code listed in Annex C.

An SPM receiving an SPDU containing a valid SI field but containing a PGI unit whose PGI code is not specified in § 8.3 or in Annex C, shall ignore that PGI unit (see notes).

An SPM reciving an SPDU containing a valid SI field but containing a PI unit whose PI code is not specified in § 8.3 or in Annex C, shall ignore that PI unit (see notes).

The SPM shall ignore any bits within a parameter field which are specified as reserved in § 8.3.

Note 1 – The received SPDU is processed as through the unknown PGI and/or PI units were not present in the SPDU.

Note 2 – The provisions permit communication with systems operating other versions of this protocol.

A.4.3.1.2 *User Data length restrictions*

If an SPM receives an SPDU, or an ordered sequence of SPDUs which together comprise a single SSDU, which contains more SS-user data than the SPM is prepared to accept (and as stated in the PICS), it shall take the actions defined in either § A.4.1.2 a) or § A.4.1.2 b).

A.4.3.2 *Invalid SPDUs*

If an invalid SPDU is received, the SPM shall:

a) take the actions defined in § A.4.1.2 a); or

b) take the actions defined in § A.4.1.2 b); or

c) take any other action that does not violate the procedures specified in this Recommendation; or

d) take no action.

A.5 *Definitions of sets and variables*

The following sets and variables are specified in this Recommendation.

A.5.1 *Functional units*

The set of all functional units specified in this Recommendation is defined as:

fu-dom = {FD, HD, EXCEP, TD, NR, SY, MA, RESYN, EX, ACT, CD}

where

FD	=	Duplex functional unit
HD	=	Half-duplex functional unit
EXCEP	=	Exceptions functional unit
TD	=	Typed data functional unit
NR	=	Negotiated release functional unit
SY	=	Minor synchronize functional unit
MA	=	Major synchronize functional unit
RESYN	=	Resynchronize functional unit
EX	=	Expedited data functional unit
ACT	=	Activity management functional unit
CD	=	Capability data functional unit

A Boolean function FU is defined over fu-dom as follows:

for f in fu-dom

FU(f) = true: if and only if the functional unit f has been selected during the session connection establishment phase.

The value is set when the ACCEPT SPDU is sent or received.

A.5.2 *Tokens*

The set of all tokens specified in this Recommendation is defined as:

tk-dom = {mi, ma, tr, dk}

where

mi	=	synchronize-minor token
ma	=	major/activity token
tr	=	release token
dk	=	data token

The following Boolean functions are defined over tk-dom:

a) AV(t), for t in tk-dom, is a function which defines the availability of the corresponding token and has the following values:

AV(mi) = FU(SY)
AV(dk) = FU(HD)
AV(tr) = FU(NR)
AV(ma) = FU(MA) or FU(ACT)

b) OWNED(t), for t in tk-dom, is a function which defines the assignment of the corresponding token and is defined as:

OWNED(t) = true: if token assigned to the SPM

OWNED(t) = false: if token not assigned to the SPM

OWNED(t) is not defined if AV(t) = false. OWNED(t) is set when:

1) the ACCEPT SPDU is sent or received; or
2) the RESYNCHRONIZE ACK SPDU is sent or received; or
3) the GIVE TOKENS SPDU is sent or received; or
4) the GIVE TOKENS CONFIRM SPDU is sent or received;
5) the ACTIVITY INTERRUPT ACK SPDU is sent or received;
6) the ACTIVITY DISCARD ACK SPDU is sent or received.

c) I(t), for t in tk-dom, is a function which, when true, indicates that the SPM has *I*nitiating rights for the behaviour controlled by the token. This applies even if the corresponding token is not available:

I(t) = ^AV(t) OR OWNED(t)

d) A(t), for t in tk-dom, is a function which, when true, indicates that the SPM has *A*ccepting rights for the behaviour controlled by the token. This applies even if the corresponding token is not available:

A(t) = ^AV(t) OR ^OWNED(t)

e) II(t), for t in tk-dom, is a function which, when true, indicates that the SPM has *I*nitiating rights as I(t), but this applies to the case when the behaviour may only be initiated if the corresponding token is available and owned:

II(t) = AV(t) AND OWNED(t)

f) AA(t), for t in tk-dom, is a function which, when true, indicates that the SPM has *A*ccepting rights as A(t), but only if the corresponding token is available, but not owned:

AA(t) = AV(t) AND ^OWNED(t)

A.5.3 *SET of tokens*

The following subsets of tk-dom are defined:

RT = {tokens requested in the input event}

GT = {tokens given in the input event}

For the purpose of the following function definitions, two further set are defined:

F = {AV, OWNED, I, A, II, AA} (the set of functions defined in § A.5.2);

S = the set of subsets of tk-dom.

The following functions are defined over F and S:

a) ALL(f, s) for f in F and s in S:

ALL(f, s) = true: all of the f(t) for t in s are true or s is empty.

For example:

ALL(A, tk-dom) = true: none of the available tokens are owned (e.g. on receipt of a FINISH SPDU).

b) ANY(f, s), for f in F and s in S:

ANY(f, s) = true: any f(t) = true for t in s when s is not empty.

For example:

ANY(II, tk-dom) = true: at least one of the available tokens is owned.

A.5.4 *Variables*

A.5.4.1 *TEXP*

TEXP is a Boolean variable having the following values:

TEXP = true: use of transport expedited service is selected for use on this session connection.

TEXP = false: use of transport expedited service is not selected for use on this session connection.

A.5.4.2 *Vact*

Vact is a Boolean variable having the following values when the activity management functional unit has been selected (FU(ACT) = true):

Vact = true:	an activity is in progress;
Vact = false:	no activity is in progress;

Vact has no defined value if FU(ACT) = false.

Vact is set as follows:

a) Vact is set false during the connection establishment phase, if the activity management functional unit has been selected (FU(ACT) = true). Otherwise, Vact is not set;

b) Vact is set true when the ACTIVITY START SPDU or ACTIVITY RESUME SPDU is sent or received (only possible when FU(ACT) = true);

c) Vact is set false when the ACTIVITY DISCARD ACK SPDU or ACTIVITY INTERRUPT ACK SPDU is sent or received;

d) Vact is set to Vnextact when a MAJOR SYNC ACK SPDU or an ACTIVITY END ACK SPDU is sent or received.

A.5.4.3 *Vnextact*

Vnextact is a Boolean variable which is used when the activity management functional unit has been selected (FU(ACT) = true). It is used to indicate the next value of Vact when a MAJOR SYNC ACK SPDU or an ACTIVITY END ACK SPDU is sent or received. Vnextact is set when a MAJOR SYNC POINT SPDU or an ACTIVITY END SPDU is sent or received:

a) Vnextact is set false if FU(ACT) = true and an ACTIVITY END SPDU is sent or received;

b) Vnextact is set true if FU(ACT) = true and a MAJOR SYNC POINT SPDU is sent or received.

Vnextact has not defined value if FU(ACT) = false.

A.5.4.4 *Vrsp and Vrspnb*

These variables are used to resolve resynchronization collisions.

Vrsp indicates what kind of resynchronization is currently in progress:

Vrsp = no	no resynchronization in progress
Vrsp = a	resynchronize abandon
Vrsp = r	resynchronize restart
Vrsp = s	resynchronize set
Vrsp = dsc	discard activity
Vrsp = int	interrupt activity

Vrspnb indicates the serial number in the case of resynchronize restart.

Vrsp and, if necessary Vrspnb, are set when a RESYNCHRONIZE SPDU, ACTIVITY INTERRUPT SPDU or an ACTIVITY DISCARD SPDU is sent or received. Vrsp is set to no when the SPM goes to STA 713.

A.5.4.5 *SPMwinner*

SPMwinner is a Boolean function which is used during resynchronization collision, that is when:

a) a RESYNCHRONIZE SPDU is received and Vrsp is not equal to no;

b) an S-RESYNCHRONIZE request is received and Vrsp is not equal to no.

The SPMwinner condition is true if the SPM (which holds the current resynchronization) wins against the new colliding event.

The SPMwinner condition is calculated as follows:

a) the next Vrsp and Vrspnb values are evaluated according to the parameters of the received event. The new calculated value for Vrsp is compared to the current Vrsp with the following ordering rule:

dsc	prevails over	int
int	prevails over	a
a	prevails over	s
s	prevails over	r

If both are equal to r, then the new calculated value for Vrspnb is compared to the current value of Vrspnb and the lower value prevails;

b) If the current value of Vrsp (and Vrspnb if necessary) prevails, then the SPMwinner condition is true (in this case, the current resynchronization prevails over the colliding one);

c) If the current value of Vrsp (and Vrspnb if necessary) does not prevail, then the SPMwinner condition is false (in this case, the colliding resynchronization prevails over the current one);

d) If the above comparison results in the equality and if the colliding event has been generated by the initiator of the session connection (either a RESYNCHRONIZE SPDU was received from the session connection initiator or a local S-RESYNCHRONIZE request was issued by the session connection initiator), then the SPMwinner condition is false.

If the SPM is winner (SPMwinner condition is true) then the current resynchronization wins against the colliding one and Vrsp and Vrspnb are not updated.

If the SPM is not winner (SPMwinner condition is false) then the colliding resynchronization is taken into account and Vrsp and Vrspnb are updated.

A.5.4.6 *Vtca*

Vtca is a Boolean variable having the following values:

Vtca = false: the SPM initiated the T-CONNECT request (transport connection initiator);

Vtca = true: the SPM received the T-CONNECT indication (transport connection acceptor).

A.5.4.7 *Vtrr*

Vtrr is a Boolean variable having the following values:

Vtrr = true: the transport connection can be reused by the SPM for another session connection;

Vtrr = false: the transport connection cannot be reused by the SPM for another session connection.

A.5.4.8 *Vcoll*

Vcoll is a Boolean variable having the following values:

Vcoll = true: a collision of FINISH SPDUs has been detected;

Vcoll = false: there has not been a collision of FINISH SPDUs.

This variable is set false during the session connection establishment phase.

A.5.4.9 *Vdnr*

Vdnr is a Boolean variable having the following values:

Vdnr = true: a DISCONNECT SPDU has been received in STA09 (following a collision of FINISH SPDUs);

Vdnr = false: no DISCONNECT SPDU has been received.

This variable is set to false during the session connection establishment phase.

A.5.4.10 *V(A)*

V(A) is used by the SPM and is the lowest serial number to which a synchronization point confirmation is expected. No confirmation is expected when V(A) = V(M).

A.5.4.11 *V(M)*

V(M) is used by the SPM and is the next serial number to be used.

A.5.4.12 *V(R)*

V(R) is used by the SPM and is the lowest serial number to which resynchronization restart is permitted.

A.5.4.13 *Vsc*

Vsc is a Boolean variable having the following values:

Vsc = true: the SS-user has the right to issue minor synchronization point responses when V(A) is less than V(M);

Vsc = false: the SS-user does not have the right to issue minor synchronization point responses.

Vsc is set false during the connection establishment phase and when a MINOR SYNC POINT SPDU is sent. Vcs is set true when a MINOR SYNC POINT SPDU is received.

Note – Table A-4/X.225 summarizes the operations on V(A), V(M), V(R) and Vsc.

TABLE A-1/X.225

Incoming Events

Abbreviated name	Category	Name and description
SACTDreq	SS-user	S-ACTIVITY-DISCARD request primitive
SACTDrsp	SS-user	S-ACTIVITY-DISCARD response primitive
SACTEreq	SS-user	S-ACTIVITY-END request primitive
SACTErsp	SS-user	S-ACTIVITY-END response primitive
SACTIreq	SS-user	S-ACTIVITY-INTERRUPT request primitive
SACTIrsp	SS-user	S-ACTIVITY-INTERRUPT response primitive
SACTRreq	SS-user	S-ACTIVITY-RESUME request primitive
SACTSreq	SS-user	S-ACTIVITY-START request primitive
SCDreq	SS-user	S-CAPABILITY-DATA request primitive
SCDrsp	SS-user	S-CAPABILITY-DATA response primitive
SCGreq	SS-user	S-CONTROL-GIVE request primitive
SCONreq	SS-user	S-CONNECT request primitive
SCONrsp+	SS-user	S-CONNECT (accept) response primitive
SCONrsp−	SS-user	S-CONNECT (reject) response primitive
SDTreq	SS-user	S-DATA request primitive
SEXreq	SS-user	S-EXPEDITED-DATA request primitive
SGTreq	SS-user	S-TOKEN-GIVE request primitive
SPTreq	SS-user	S-TOKEN-PLEASE request primitive
SRELreq	SS-user	S-RELEASE request primitive
SRELrsp+	SS-user	S-RELEASE (accept) response primitive
SRELrsp−	SS-user	S-RELEASE (reject) response primitive
SRSYNreq(a)	SS-user	S-RESYNCHRONIZE (abandon) request primitive
SRSYNreq(r)	SS-user	S-RESYNCHRONIZE (restart) request primitive
SRSYNreq(s)	SS-user	S-RESYNCHRONIZE (set) request primitive
SRSYNrsp	SS-user	S-RESYNCHRONIZE response primitive
SSYNMreq	SS-user	S-SYNC-MAJOR request primitive
SSYNMrsp	SS-user	S-SYNC-MAJOR response primitive
SSYNmreq	SS-user	S-SYNC-MINOR request primitive

TABLE A-1/X.225 *(cont.)*

Abbreviated name	Category	Name and description
SSYNmrsp	SS-user	S-SYNC-MINOR response primitive
STDreq	SS-user	S-TYPED-DATA request primitive
SUABreq	SS-user	S-U-ABORT request primitive
SUERreq	SS-user	S-U-EXCEPTION-REPORT request primitive
TCONind	TS-provider	T-CONNECT indication primitive
TCONcnf	TS-provider	T-CONNECT confirm primitive
TDISind	TS-provider	T-DISCONNECT indication primitive
TIM	Timer	Time out
AA	SPDU	ABORT ACCEPT SPDU
AB-nr	SPDU	ABORT (not reuse) SPDU
AB-r	SPDU	ABORT (reuse) SPDU
AC	SPDU	ACCEPT SPDU (see Note 1)
AD	SPDU	ACTIVITY DISCARD SPDU
ADA	SPDU	ACTIVITY DISCARD ACK SPDU
AE	SPDU	ACTIVITY END SPDU
AEA	SPDU	ACTIVITY END ACK SPDU
AI	SPDU	ACTIVITY INTERRUPT SPDU
AIA	SPDU	ACTIVITY INTERRUPT ACK SPDU
AR	SPDU	ACTIVITY RESUME SPDU
AS	SPDU	ACTIVITY START SPDU
CD	SPDU	CAPABILITY DATA SPDU
CDA	SPDU	CAPABILITY DATA ACK SPDU
CDO	SPDU	CONNECT DATA OVERFLOW SPDU
CN	SPDU	CONNECT SPDU
DN	SPDU	DISCONNECT SPDU
DT	SPDU	DATA TRANSFER SPDU
ED	SPDU	EXCEPTION DATA SPDU
ER	SPDU	EXCEPTION REPORT SPDU
EX	SPDU	EXPEDITED DATA SPDU
FN-nr	SPDU	FINISH (not reuse) SPDU
FN-r	SPDU	FINISH (reuse) SPDU
GT	SPDU	GIVE TOKENS SPDU with Token Item parameter (see Note 2)
GTA	SPDU	GIVE TOKENS ACK SPDU
GTC	SPDU	GIVE TOKENS CONFIRM SPDU
MAA	SPDU	MAJOR SYNC ACK SPDU
MAP	SPDU	MAJOR SYNC POINT SPDU
MIA	SPDU	MINOR SYNC ACK SPDU

TABLE A-1/X.225 *(end)*

Abbreviated name	Category	Name and description
MIP	SPDU	MINOR SYNC POINT SPDU
NF	SPDU	NOT FINISHED SPDU
OA	SPDU	OVERFLOW ACCEPT SPDU
PR-AB	SPDU	PREPARE (ABORT) SPDU
PR-MAA	SPDU	PREPARE (MAJOR SYNC ACK) SPDU
PR-RA	SPDU	PREPARE (RESYNCHRONIZE ACK) SPDU
PR-RS	SPDU	PREPARE (RESYNCHRONIZE) SPDU
PT	SPDU	PLEASE TOKENS SPDU with Token Item parameter (see Notes 1 and 2)
RA	SPDU	RESYNCHRONIZE ACK SPDU
RF-nr	SPDU	REFUSE (not reuse) SPDU
RF-r	SPDU	REFUSE (reuse) SPDU
RS-a	SPDU	RESYNCHRONIZE (abandon) SPDU
RS-r	SPDU	RESYNCHRONIZE (restart) SPDU
RS-s	SPDU	RESYNCHRONIZE (set) SPDU
TD	SPDU	TYPED DATA SPDU

Note 1 – If the Token Item parameter is present in the ACCEPT SPDU, both the AC event and the PT event occur.

Note 2 – GIVE TOKENS SPDU without Token Item parameter and PLEASE TOKENS SPDU without Token Item parameter are used to herald a concatenated sequence of SPDUs Concatenation of SPDUs and separation of TSDUs are not handled by the state tables.

TABLE A-2/X.225

States

Abbreviated name	Name and description
STA 01	Idle, no transport connection
STA 01A	Wait for the ABORT ACCEPT SPDU
STA 01B	Wait for T-CONNECT confirm
STA 01C	Idle, transport connected
STA 01D	Wait for the CONNECT DATA OVERFLOW SPDU
STA 02A	Wait for the ACCEPT SPDU
STA 02B	Wait for the OVERFLOW ACCEPT SPDU
STA 03	Wait for the DISCONNECT SPDU
STA 04A	Wait for the MAJOR SYNC ACK SPDU or PREPARE (MAJOR SYNC ACK) SPDU
STA 04B	Wait for the ACTIVITY END ACK SPDU or PREPARE (MAJOR SYNC ACK) SPDU
STA 05A	Wait for the RESYNCHRONIZE ACK SPDU or PREPARE (RESYNCHRONIZE ACK) SPDU
STA 05B	Wait for the ACTIVITY INTERRUPT ACK SPDU or PREPARE (RESYNCHRONIZE ACK) SPDU
STA 05C	Wait for the ACTIVITY DISCARD ACK SPDU or PREPARE (RESYNCHRONIZE ACK) SPDU
STA 06	Wait for the RESYNCHRONIZE SPDU [resynchronization collision after receiving PREPARE (RESYNCHRONIZE) SPDU]
STA 08	Wait for S-CONNECT response
STA 09	Wait for S-RELEASE response
STA 10A	Wait for S-SYNC-MAJOR response
STA 10B	Wait for S-ACTIVITY-END response
STA 11A	Wait for S-RESYNCHRONIZE response
STA 11B	Wait for S-ACTIVITY-INTERRUPT response
STA 11C	Wait for S-ACTIVITY-DISCARD response
STA 15A	After PREPARE, wait for the MAJOR SYNC ACK SPDU or the ACTIVITY END ACK SPDU
STA 15B	After PREPARE, wait for the RESYNCHRONIZE SPDU or the ACTIVITY INTERRUPT SPDU or the ACTIVITY DISCARD SPDU
STA 15C	After PREPARE, wait for the RESYNCHRONIZE ACK SPDU or the ACTIVITY INTERRUPT ACK SPDU or the ACTIVITY DISCARD ACK SPDU
STA 15D	After PREPARE, wait for the ABORT SPDU
STA 16	Wait for T-DISCONNECT indication
STA 18	Wait for the GIVE TOKENS ACK SPDU
STA 19	Wait for a recovery request or SPDU (initiator of EXCEPTION DATA SPDU)
STA 20	Wait for a recovery SPDU or request
STA 21	Wait for the CAPABILITY DATA ACK SPDU
STA 22	Wait for S-CAPABILITY-DATA response
STA 713	Data transfer state

TABLE A-3/X.225

Outgoing Events

Abbreviated name	Category	Name and description
SACTDind	SS-provider	S-ACTIVITY-DISCARD indication primitive
SACTDcnf	SS-provider	S-ACTIVITY-DISCARD confirm primitive
SACTEind	SS-provider	S-ACTIVITY-END indication primitive
SACTEcnf	SS-provider	S-ACTIVITY-END confirm primitive
SACTIind	SS-provider	S-ACTIVITY-INTERRUPT indication primitive
SACTIcnf	SS-provider	S-ACTIVITY-INTERRUPT confirm primitive
SACTRind	SS-provider	S-ACTIVITY-RESUME indication primitive
SACTSind	SS-provider	S-ACTIVITY-START indication primitive
SCDind	SS-provider	S-CAPABILITY-DATA indication primitive
SCDcnf	SS-provider	S-CAPABILITY-DATA confirm primitive
SCGind	SS-provider	S-CONTROL-GIVE indication primitive
SCONind	SS-provider	S-CONNECT indication primitive
SCONcnf+	SS-provider	S-CONNECT (accept) confirm primitive
SCONcnf−	SS-provider	S-CONNECT (reject) confirm primitive
SDTind	SS-provider	S-DATA indication primitive
SEXind	SS-provider	S-EXPEDITED-DATA indication primitive
SGTind	SS-provider	S-TOKEN-GIVE indication primitive
SPABind	SS-provider	S-P-ABORT indication primitive
SPERind	SS-provider	S-P-EXCEPTION-REPORT indication primitive
SPTind	SS-provider	S-TOKEN-PLEASE indication primitive
SRELind	SS-provider	S-RELEASE indication primitive
SRELcnf+	SS-provider	S-RELEASE (accept) confirm primitive
SRELcnf−	SS-provider	S-RELEASE (reject) confirm primitive
SRSYNind	SS-provider	S-RESYNCHRONIZE indication primitive
SRSYNcnf	SS-provider	S-RESYNCHRONIZE confirm primitive
SSYNMind	SS-provider	S-SYNC-MAJOR indication primitive
SSYNMcnf	SS-provider	S-SYNC-MAJOR confirm primitive
SSYNmind	SS-provider	S-SYNC-MINOR indication primitive
SSYNmcnf	SS-provider	S-SYNC-MINOR confirm primitive

TABLEAU A-3/X.225 *(cont.)*

Abbreviated name	Category	Name and description
STDind	SS-provider	S-TYPED-DATA indication primitive
SUABind	SS-provider	S-U-ABORT indication primitive
SUERind	SS-provider	S-U-EXCEPTION-REPORT indication primitive
TCONreq	TS-user	T-CONNECT request primitive
TCONrsp	TS-user	T-CONNECT response primitive
TDISreq	TS-user	T-DISCONNECT request primitive
AA	SPDU	ABORT ACCEPT SPDU
AB-nr	SPDU	ABORT (not reuse) SPDU
AB-r	SPDU	ABORT (reuse) SPDU
AC	SPDU	ACCEPT SPDU
AD	SPDU	ACTIVITY DISCARD SPDU
ADA	SPDU	ACTIVITY DISCARD ACK SPDU
AE	SPDU	ACTIVITY END SPDU
AEA	SPDU	ACTIVITY END ACK SPDU
AI	SPDU	ACTIVITY INTERRUPT SPDU
AIA	SPDU	ACTIVITY INTERRUPT ACK SPDU
AR	SPDU	ACTIVITY RESUME SPDU
AS	SPDU	ACTIVITY START SPDU
CD	SPDU	CAPABILITY DATA SPDU
CDA	SPDU	CAPABILITY DATA ACK SPDU
CDO	SPDU	CONNECT DATA OVERFLOW SPDU
CN	SPDU	CONNECT SPDU
DN	SPDU	DISCONNECT SPDU
DT	SPDU	DATA TRANSFERT SPDU
ED	SPDU	EXCEPTION DATA SPDU
EX	SPDU	EXPEDITED DATA SPDU
FN-nr	SPDU	FINISH (not reuse) SPDU
FN-r	SPDU	FINISH (reuse) SPDU
GT	SPDU	GIVE TOKENS SPDU
GTA	SPDU	GIVE TOKENS ACK SPDU
GTC	SPDU	GIVE TOKENS CONFIRM SPDU
MAA	SPDU	MAJOR SYNC ACK SPDU
MAP	SPDU	MAJOR SYNC POINT SPDU
MIA	SPDU	MINOR SYNC ACK SPDU
MIP	SPDU	MINOR SYNC POINT SPDU
NF	SPDU	NOT FINISHED SPDU
OA	SPDU	OVERFLOW ACCEPT SPDU
PR-MAA	SPDU	PREPARE (MAJOR SYNC ACK) SPDU
PR-AB	SPDU	PREPARE (ABORT) SPDU

TABLE A-3/X.225 *(end)*

Abbreviated name	Category	Name and description
PR-RA	SPDU	PREPARE (RESYNCHRONIZE ACK) SPDU
PR-RS	SPDU	PREPARE (RESYNCHRONIZE) SPDU
PT	SPDU	PLEASE TOKENS SPDU
RA	SPDU	RESYNCHRONIZE ACK SPDU
RF-nr	SPDU	REFUSE (not reuse) SPDU
RF-r	SPDU	REFUSE (reuse) SPDU
RS-a	SPDU	RESYNCHRONIZE (abandon) SPDU
RS-r	SPDU	RESYNCHRONIZE (restart) SPDU
RS-s	SPDU	RESYNCHRONIZE (set) SPDU
TD	SPDU	TYPED DATA SPDU

TABLE A-4/X.225

Operations on variable

Events	Condition for valid SPDU or primitive	Condition for update of variables	Operations on variables			
			V(A)	V(M)	V(R)	Vsc
SSYNMreq SSYNmreq SACTEreq		if Vsc true	set to V(M)	V(M) + 1	unchanged	false
		if Vsc false	unchanged	V(M) + 1	unchanged	false
MAP SPDU AE SPDU	sn = V(M)	if Vsc true	unchanged	V(M) + 1	unchanged	unchanged
		if Vsc false	set to V(M)	V(M) + 1	unchanged	unchanged
MIP SPDU	sn = V(M)	if Vsc true	unchanged	V(M) + 1	unchanged	true
		if Vsc false	set to V(M)	V(M) + 1	unchanged	true
SSYNMrsp SACTErsp MAA SPDU AEA SPDU	sn = V(M) – 1		set to V(M)	unchanged	set to V(M)	unchanged
SSYNmrsp	Vsc = true and V(M) > sn >= V(A) *		set to ns + 1	unchanged	unchanged	unchanged
MIA SPDU	Vsc = false and V(M) > sn >= V(A) *		set to sn + 1	unchanged	unchanged	unchanged
SRSYNreq	a: not applicable	abandon	unchanged	unchanged	unchanged	unchanged
	r: V(M) >= sn >= V(R)	restart	unchanged	unchanged	unchanged	unchanged
	s: sn <= 999 999	set	unchanged	unchanged	unchanged	unchanged
RS SPDU	a: sn <= 999 999	abandon	unchanged	max (sn, V(M))	unchanged	unchanged
	r: sn >= V(R)	restart	unchanged	unchanged	unchanged	unchanged
	s: sn <= 999 999	set	unchanged	unchanged	unchanged	unchanged
SRSYNrsp	a: sn = V(M)	abandon	set to sn	set to sn	0	unchanged
	r: sn as in RS SPDU	restart	set to sn	set to sn	unchanged	unchanged
	s: sn <= 999 999	set	set to sn	set to sn	0	unchanged
RA SPDU	a: sn >= V(M)	abandon	set to sn	set to sn	0	unchanged
	r: sn as in RS SPDU	restart	set to sn	set to sn	unchanged	unchanged
	s: sn <= 999 999	set	set to sn	set to sn	0	unchanged
SACTRreq AR SPDU			set to sn + 1	set to sn + 1	set to 1	unchanged
SACTSreq AS SPDU			set to 1	set to 1	set to 1	unchanged
SCONrsp AC SPDU		sn present	set to sn	set to sn	0	false

sn: synchronization point serial number quoted in SS-user request or SPDU
>=: greater than or equal to
<=: less than or equal to
*: synchronization point serial number not equal to V(M) – 1 if major synchronization or activity end outstanding

TABLE A-5/X.225

Specific actions

[1]	Set Vtca = true
[2]	Set Vtca = false
[3]	Stop timer TIM
[4]	Start timer TIM
[5]	Set V(A) = V(M) = serial number in ACCEPT SPDU
	Set V(R) = 0
	Set Vcoll = false
	Set Vrsp = no
	Set Vsc = false
	Set TEXP
	Set FU(f) for f in fu-dom according to the intersection of Session User Requirements in the CONNECT SPDU and Session User Requirements in the ACCEPT SPDU
	If FU(ACT) = true, set Vact = false
	Set Vdnr = false
[6]	Recall the queued events until the queue is empty
[7]	Set Vtrr = true
[8]	Set Vtrr = false
[9]	Set Vtrr according to the Transport Disconnect PV field in the SPDU. As a local decision, Vtrr may always be set false
[10]	Store the event in the queue
[11]	Update the position of the tokens
[12]	Set Vact = true
[13]	Set Vnextact
[14]	Set Vact = Vnextact
[15]	Clear the queue
[16]	Update Vrsp and, if RS-r, Vrspnb
[17]	Not used
[18]	Set Vcoll = true
[19]	V(M) = maximum [V(M), received serial number]
[20]	Set Vsc = false
[21]	Set V(M) = V(M) + 1
[22]	Set V(R) = V(A) = V(M)
[23]	If Vsc = false, set V(A) = V(M). Set Vsc = true
	Set V(M) = V(M) + 1
[24]	If Vsc = true, set V(A) = V(M). Set Vsc = false
	Set V(M) = V(M) + 1
[25]	Set V(A) = serial number + 1
[26]	Set V(A) = V(M) = V(R) = 1
[27]	Set V(A) = V(M) = serial number + 1
	Set V(R) = 1
[28]	Set V(A) = V(M) = serial number
	If Vrsp = a, then set V(R) = 0
	If Vrsp = s, then set V(R) = 0
	Set Vrsp = no
[29]	Set the position of the tokens such that all available tokens are owned. Set Vact = false
	Set Vrsp = no
[30]	Set the position of the tokens such that all available tokens are not owned. Set Vact = false
	Set Vrsp = no
[31]	If Vsc = false, set V(A) = V(M)
	Set V(M) = V(M) + 1
[32]	Set Vdnr = true
[50]	Preserve user data for subsequent SCONind
[51]	If p201 send subsequent CDO SPDUs until ¬p201

TABLE A-6/X.225

Predicates

p01	^Vtca
p02	local choice & ^TEXP
p03	I(dk)
p04	FU(FD) & ^Vcoll
p05	A(dk)
p06	FU(TD)
p07	FU(TD) & ^Vcoll
p08	FU(EX)
p09	FU(EX) & ^Vcoll
p10	^Vcoll
p11	II(ma)
p12	(^FU(ACT) OR Vact) & A(dk) & A(mi) & AA(ma)
p13	(^FU(ACT) OR Vact) & I(dk) & I(mi) & II(ma)
p14	(^FU(ACT) OR Vact) & A(dk) & AA(mi)
p15	(^FU(ACT) OR Vact) & I(dk) & II(mi)
p16	^TEXP
p17	(^FU(ACT) OR Vact) & FU(SY) & ^Vsc
p18	(^FU(ACT) OR Vact) & FU(SY) & Vsc
p19	serial number = V(M)
p20	serial number = V(M) – 1
p21	V(M) > serial number > = V(A)
p22	Unused
p23	FU(ACT) & ^Vnextact
p24	^SPMwinner
p25	(FU(SY) OR FU(MA)) & FU(RESYN)
p26	(^FU(ACT) OR Vact)
p27	Vrsp = no
p28	FU(RESYN)
p29	(^FU(ACT) OR Vact) & FU(RESYN)
p30	(^FU(ACT) OR Vnextact)
p31	FU(ACT) & Vnextact
p32	serial number > = V(R)
p33	V(M) > = serial number > = V(R)
p34	FU(ACT)
p35	FU(RESYN) & ^TEXP
p36	FU(RESYN) & TEXP
p37	FU(ACT) & TEXP
p38	FU(ACT) & ^TEXP
p39	Vact & II(ma)
p40	AA(ma)
p41	Vrsp = dsc
p42	Vrsp = int
p43	((Vrsp = r) & (serial number = Vrspnb)) OR ((Vrsp = a) & (serial number = V(M))) OR (Vrsp = s)
p44	(FU(ACT) & ^Vact) & A(dk) & A(mi) & A(ma)
p45	(FU(ACT) & ^Vact) & I(dk) & I(mi) & I(ma)
p46	FU(CD) & (FU(ACT) & ^Vact) & A(dk) & A(mi) & ^OWNED(ma)
p47	FU(CD) & (FU(ACT) & ^Vact) & I(dk) & I(mi) & OWNED(ma)
p48	FU(EXCEP) & FU(HD)
p49	((Vrsp = r) & (serial number = Vrspnb)) OR ((Vrsp = a) & (serial number > = V(M))) OR (Vrsp = s)
p50	FU(EXCEP) & (^FU(ACT) OR Vact) & AA(dk)
p51	FU(EXCEP) & (^FU(ACT) OR Vact) & II(dk)
p52	FU(EXCEP) & ^FU(ACT) & II(dk)
p53	ANY(AV, RT)

TABLE A-6/X.225 *(cont.)*

p54	ALL(I, GT) & ANY (AV, GT)
p55	(FU(ACT) & ˆVact) & ALL(I, tk-dom)
p56	Unused
p57	ALL(I, GT) & (dk not in GT) & ANY (AV, GT)
p58	ALL(I, GT) & (dk in GT)
p59	ALL(A, GT) & ANY (AV, GT)
p60	ALL(A, GT) & (dk not in GT) & ANY (AV, GT)
p61	ALL(A, GT) & (dk in GT)
p62	(FU(ACT) & ˆVact) & ALL(A, tk-dom)
p63	ALL(I, tk-dom) & (ˆFU(ACT) OR ˆVact)
p64	local choice & ˆVtca & ˆTEXP
p65	ANY(AV, tk-dom)
p66	Vtrr
p67	FU(NR)
p68	ALL (A, tk-dom) & (ˆFU(ACT) OR ˆVact)
p69	Vcoll
p70	FU(FD)
p71	FU(ACT) & Vact & I(dk) & I(mi) & II(ma)
p72	FU(ACT) & Vact & A(dk) & A(mi) & AA(ma)
p75	(Vcoll & Vdnr) OR ˆVcoll
p76	CN SPDU is not acceptable to the SPM for transient or persistent reason (see § 8.3.5.9)
p201	More user data to send
p202	End of user data
p204	More than 10 240 octets of SS-user data to be transferred

A.6 *Notes to Tables A-7/X.225 to A-15/X.225:*

Note 1 – PR is not sent if TEXP is false.

Note 2 – The serial number given in the indication is V(M).

Note 3 – SxABind means generate event SUABind if bit 2 of the Transport Disconnect PV field in the ABORT SPDU has the value "user abort". Otherwise, SxABind means generate the event SPABind.

Note 4 – PR-AB is only sent if TEXP is true and the SS-user data exceeds 9 octets (§ 7.9.2).

TABLE A-7/X.225

Connection establishment state table

STATE EVENT	STA01 idle no TC	STA01A await AA	STA01B await TCONcnf	STA01C idle TC con	STA01D await CDO	STA02A await AC	STA02B await OA	STA08 await SCONrsp	STA15D await after PR-AB	STA16 await TDISind
AC	/ /	STA01A	/ /	TDISreq STA01		SCONcnf+ [5] [11] STA713 [6]			STA15D	STA16
CDO	/ /		/ /	TDISreq STA01	^p202 [50] STA01D p202 SCONind STA08				STA15D	
CN	/ /	TDISreq [3] STA01	/ /	^p01 & ^p76 & p204 OA [50] STA01D ^p01 & ^p76 & ^p204 SCONind STA08 ^p01 & p76 & ^p02 RF-nr [4] STA16 ^p01 & p76 & p02 RF-r STA01C p01 TDISreq STA01						TDISreq [3] STA01
OA	/ /		/ /	TDISreq STA01			CDO [51] STA02A		STA15D	
RF-nr	/ /	STA01A	/ /	TDISreq STA01		SCONcnf– TDISreq STA01	SCONcnf– TDISreq STA01			STA16

TABLE A-7/X.225 *(cont.)*

STATE / EVENT	STA01 idle no TC	STA01A await AA	STA01B await TCONcnf	STA01C idle TC con	STA01D await CDO	STA02A await AC	STA02B await OA	STA08 await SCONrsp	STA15D wait after PR-AB	STA16 await TDISind
RF-r	/ /	STA01A	/ /	TDISreq STA01		ˆp02 SCONcnf– TDISreq STA01 p02 SCONcnf– STA01C	ˆp02 SCONcnf– TDISreq STA01 p02 SCONcnf– STA01C			STA16
SCONreq	TCONreq [2] STA01B			p01 & p204 CN STA02B p01 & ˆp204 CN STA02A						
SCONrsp+								AC [5] [11] STA713	STA15D	
SCONrsp–								ˆp02 RF-nr [4] STA16 p02 RF-r STA01C	STA15D	
TCONcnf	/ /	/ /	p204 CN STA02B ˆp204 CN STA02A	/ /	/ /	/ /	/ /	/ /	/ /	/ /
TCONind	TCONrsp [1] STA01C	/ /	/ /	/ /	/ /	/ /	/ /	/ /	/ /	/ /

TABLE A-8/X.225

Data transfer state table

STATE EVENT	STA01A await AA	STA01C idle TC con	STA01D await CDO	STA02A await AC	STA03 await DN	STA04A await PR or MAA	STA04B await PR or AEA	STA05A await PR or RA	STA05B await PR or AIA
DT	STA01A	TDISreq STA01	TDISreq STA01		p05&p10 SDTind STA03	p05 SDTind STA04A	p05 SDTind STA04B	p05 STA05A	p05 STA05B
EX	STA01A	TDISreq STA01	TDISreq STA01	[10] STA02A	p09 SEXind STA03	p08 SEXind STA04A	p08 SEXind STA04B	p08 STA05A	p08 STA05B
TD	STA01A	TDISreq STA01	TDISreq STA01		p06&p10 STDind STA03	p06 STDind STA04A	p06 STDind STA04B	p06 STA05A	p06 STA05B
SDTreq									
SEXreq									
STDreq									

TABLE A-8/X.225 *(cont.)*

Data transfer state table

STATE EVENT	STA05C await PR or ADA	STA06 await RS after coll	STA09 await SRELrsp	STA10A await SSYNMrsp	STA10B await SACTErsp	STA15A wait after PR-MAA	STA15B wait after PR-RS
DT	p05 STA05C	p05 STA06				p05 SDTind STA15A	p05 STA15B
EX	p08 STA05C	p08 [10] STA06				p08 [10] STA15A	
TD	p06 STA05C	p06 STA06				p06 STDind STA15A	p06 STA15B
SDTreq			p04 DT STA09	p03 DT STA10A	p03 DT STA10B		p03 STA15B
SEXreq			p09 EX STA09	p08 EX STA10A	p08 EX STA10B		p08 STA15B
STDreq			p07 TD STA09	p06 TD STA10A	p06 TD STA10B		p06 STA15B

TABLE A-8/X.225 *(end)*

Data transfer state table

STATE / EVENT	STA15C wait after PR-RA	STA15C wait after PR-AB	STA16 await TDISind	STA18 await GTA	STA19 await recovery (init)	STA20 await recovery	STA21 await CDA	STA713 data transfer
DT	p05 STA15C	STA15D	STA16	p70 SDTind STA18	STA19	p05 STA20	p70 SDTind STA21	p05 SDTind STA713
EX	p08 [10] STA15C		STA16	p08 SEXind STA18	p08 STA19	p08 STA20	p08 SEXind STA21	p08 SEXind STA713
TD	p06 STA15C	STA15D	STA16	p06 STDind STA18	p06 STA19	p06 STA20	p06 STDind STA21	p06 STDind STA713
SDTreq		STA15D		p70 DT STA18				p03 DT STA713
SEXreq		STA15D		p08 EX STA18				p08 EX STA713
STDreq		STA15D		p06 TD STA18				p06 TD STA713

TABLE A-9/X.225

Synchronization state table

STATE / EVENT	STA01A await AA	STA01C idle CT con	STA01D await CDO	STA04A await PR or MAA	STA04B await PR or AEA	STA05A await PR or RA	STA05B await PR or AIA	STA05C await PR or ADA
MAA or AEA	STA01A	TDISreq STA01	TDISreq STA01	p16&p20 SSYNMcnf [14] [22] STA713	p16&p20 SACTEcnf [14] [22] STA713	STA05A	STA05B	STA05C
MAP	STA01A	TDISreq STA01	TDISreq STA01			p12 STA05A		
PR-MAA	STA01A	TDISreq STA01	TDISreq STA01	STA15A	STA15A	STA05A	STA05B	STA05C
SSYNMreq								
SSYNMrsp								

TABLE A-9/X.225 *(cont.)*

Synchronization state table

STATE EVENT	STA06 await RS after coll	STA10A await SSYNMrsp	STA15A wait after PR-MAA	STA15B wait after PR-RS	STA15C wait after PR-RA	STA15C wait after PR-AB
MAA or AEA	STA06		p20&p23 SSYNMcnf [14] [22] STA713 [6] p20&p23 SACTEcnf [14] [22] STA713 [6]	STA15B	STA15C	STA15D
MAP	p12 STA06			p12 STA15B	p12 STA15C	STA15D
PR-MAA						
SSYNMreq				p13 STA15B		STA15D
SSYNMrsp		PR-MAA (1) MAA [14] [22] STA713		STA15B		STA15D

TABLE A-9/X.225 *(cont.)*

Synchronization state table

STATE EVENT	STA16 await TDISind	STA19 await recovery (init)	STA20 await recovery	STA713 data transfer
MAA or AEA	STA16		p20 STA20	
MAP	STA16	p12&p19 [31] STA19	p12&p19 [31] STA20	p12&p19 SSYNMind [13] [31] STA10A
PR-MAA	STA16			
SSYNMreq				p13 MAP [13] [24] STA04A
SSYNMrsp				

TABLE A-9/X.225 *(cont.)*

Synchronization state table

STATE EVENT	STA01A await AA	STA01C idle TC con	STA01D await CDO	STA03 await DN	STA04A await PR or MAA	STA04B await PR or AEA	STA05A await PR or RA
AE	STA01A	TDISreq STA01	TDISreq STA01				p72 STA05A
MIA	STA01A	TDISreq STA01	TDISreq STA01	p17&p21 SSYNmcnf [25] STA03	p17&^p20&p21 SSYNmcnf [25] STA04A	p17&^p20&p21 SSYNmcnf [25] STA04B	p17 STA05A
MIP	STA01A	TDISreq STA01	TDISreq STA01				p14 STA05A
SACTEreq							
SACTErsp							
SSYNmreq							
SSYNmrsp							

TABLE A-9/X.225 *(cont.)*

Synchronization state table

STATE EVENT	STA05B await PR or AIA	STA05C await PR or ADA	STA06 await RS after coll	STA09 await SRELrsp	STA10A await SSYNMrsp
AE			p72 STA06		
MIA	p17 STA05B	p17 STA05C	p17 STA06		
MIP	p14 STA05B	p14 STA05C	p14 STA06		
SACTEreq					
SACTErsp					
SSYNmreq					
SSYNmrsp				p18&p21 MIA [25] STA09	p18&^p20&p21 MIA [25] STA10A

TABLE A-9/X.225 *(cont.)*

Synchronization state table

STATE EVENT	STA10B await SACTErsp	STA15A wait after PR-MAA	STA15B wait after PR-RS	STA15C wait after PR-RA	STA15C wait after PR-AB	STA16 await TDISind
AE			p72 STA15B	p72 STA15C	STA15D	STA16
MIA		p17&ˆp20&p21 SSYNmcnf [25] STA15A	p17 STA15B	p17 STA15C	STA15D	STA16
MIP			p14 STA15B	p14 STA15C	STA15D	STA16
SACTEreq			p71 STA15B		STA15D	
SACTErsp	PR-MAA (1) AEA [14] [22] STA713				STA15D	
SSYNmreq			p15 STA15B		STA15D	
SSYNmrsp	p18&ˆp20&p21 MIA [25] STA10B		p18&p21 STA15B		STA15D	

TABLE A-9/X.225 *(end)*

Synchronization state table

STATE / EVENT	STA19 await recovery (init)	STA20 await recovery	STA713 data transfer
AE	p72&p19 [31] STA19	p72&p19 [31] STA20	p72&p19 SACTEind [13] [31] STA10B
MIA	p17&p21 [25] STA19	p17&p21 STA20	p17&p21 SSYNmcnf [25] STA713
MIP	p14&p19 [23] STA19	p14&p19 [23] STA20	p14&p19 SSYNmind [23] STA713
SACTEreq			p71 AE [13] [24] STA04B
SACTErsp			
SSYNmreq			p15 MIP [24] STA713
SSYNmrsp			p18&p21 MIA [25] STA713

TABLE A-10/X.225

Resynchronization state table

STATE EVENT	STA01A await AA	STA01C idle TC con	STA01D await CDO	STA02A await AC	STA03 await DN	STA04A await PR or MAA	STA04B await PR or AEA
PR-RA	STA01A	TDISreq STA01	TDISreq STA01				
PR-RS	STA01A	TDISreq STA01	TDISreq STA01	[10] STA02A	p10 STA15B	STA15B	STA15B
RA	STA01A	TDISreq STA01	TDISreq STA01				
RS-a	STA01A	TDISreq STA01	TDISreq STA01		p10&^p34&p35 [19] SRSYNind(2) [16] STA11A	p35 [19] SRSYNind(2) [16] STA11A	p35 [19] SRSYNind(2) [16] STA11A
RS-r	STA01A	TDISreq STA01	TDISreq STA01		p10&^p34& p35&p32 SRSYNind [16] STA11A	p32&p35 SRSYNind [16] STA11A	p32&p35 SRSYNind [16] STA11A
RS-s	STA01A	TDISreq STA01	TDISreq STA01		p10&^p34&p35 SRSYNind [16] STA11A	p35 SRSYNind [16] STA11A	p35 SRSYNind [16] STA11A
SRSYNreq(a)						p28 PR-RS(1) RS-a [16] STA05A	
SRSYNreq(r)							
SRSYNreq(s)						p28 PR-RS(1) RS-s [16] STA05A	
SRSYNrsp							

TABLE A-10/X.225 *(cont.)*

Resynchronization state table

STATE EVENT	STA05A await PR or RA	STA05B await PR or AIA	STA05C await PR or ADA	STA06 await RS after coll	STA09 await SRELrsp
PR-RA	STA15C	STA15C	STA15C	[10] STA06	
PR-RS	STA06	STA05B	STA05C	[10] STA06	
RA	p35&p49 SRSYNcnf [28] [11] STA713				
RS-a	^p24&p35 STA05A p24&p35 [19] SRSYNind(2) [16] STA11A	p28 STA05B	p28 STA05C	^p24 STA05A [6] p24 [19] SRSYNind(2) [16] STA11A [6]	
RS-r	^p24&p32&p35 STA05A p24&p32&p35 SRSYNind [16] STA11A	p28 STA05B	p28 STA05C	^p24&p32 STA05A [6] p24&p32 SRSYNind [16] STA11A [6]	
RS-s	^p24&p35 STA05A p24&p35 SRSYNind [16] STA11A	p28 STA05B	p28 STA05C	^p24 STA05A [6] p24 SRSYNind [16] STA11A [6]	
SRSYNreq(a)					p10&p28&^p34 PR-RS(1) RS-a [16] STA05A
SRSYNreq(r)					p10&p25&^p34&p33 PR-RS(1) RS-r [16] STA05A
SRSYNreq(s)					p10&p25&^p34 PR-RS(1) RS-s [16] STA05A
SRSYNrsp					

TABLE A-10/X.225 *(cont.)*

Resynchronization state table

STATE EVENT	STA10A await SSYNMrsp	STA10B await SACTErsp	STA11A await SRSYNrsp	STA15A wait after PR-MAA	STA15B wait after PR-RS	STA15C wait after PR-RA	STA15D wait after PR-AB
PR-RA							
PR-RS	STA15B	STA15B		[10] STA15A		[10] STA15C	
RA						p36&p49 SRSYNcnf [28] [11] STA713 [6]	STA15D
RS-a	p35 [19] SRSYNind(2) [16] STA11A				p29 [19] SRSYNind(2) [16] STA11A		STA15D
RS-r					p32&p29 SRSYNind [16] STA11A		STA15D
RS-s	p35 SRSYNind [16] STA11A				p29 SRSYNind [16] STA11A		STA15D
SRSYNreq(a)	p28 PR-RS(1) RS-a [16] STA05A	p28 PR-RS(1) RS-a [16] STA05A	p24 PR-RS(1) RS-a [16] STA05A	p28&p30 PR-RS(1) RS-a [16] STA05A [6]	p27&p28 PR-RS(1) RS-a [16] STA06		STA15D
SRSYNreq(r)	p25&p33 PR-RS(1) RS-r [16] STA05A	p25&p33 PR-RS(1) RS-r [16] STA05A	p24&p33 PR-RS(1) RS-r [16] STA05A		p25&p27&p33 PR-RS(1) RS-r [16] STA06		STA15D
SRSYNreq(s)	p25 PR-RS(1) RS-s [16] STA05A	p25 PR-RS(1) RS-s [16] STA05A	p24 PR-RS(1) RS-s [16] STA05A	p28&p30 PR-RS(1) RS-s [16] STA05A [6]	p25&p27 PR-RS(1) RS-s [16] STA06		STA15D
SRSYNrsp			p43 PR-RA(1) RA [28] [11] STA713				STA15D

TABLE A-10/X.225 *(end)*

Resynchronization state table

STATE EVENT	STA16 await TDISind	STA18 await GTA	STA19 await recovery (init)	STA20 await recovery	STA713 data transfer
PR-RA	STA16				
PR-RS	STA16	[10] STA18	STA15B	STA15B	p26 STA15B ^p26 [10] STA713
RA	STA16				
RS-a	STA16		p35 [19] SRSYNind(2) [16] STA11A	p35 [19] SRSYNind(2) [16] STA11A	p26&p35 [19] SRSYNind(2) [16] STA11A
RS-r	STA16		p32&p35 SRSYNind [16] STA11A	p32&p35 SRSYNind [16] STA11A	p32&p26&p35 SRSYNind [16] STA11A
RS-s	STA16		p35 SRSYNind [16] STA11A	p35 SRSYNind [16] STA11A	p26&p35 SRSYNind [16] STA11A
SRSYNreq(a)				p28 PR-RS(1) RS-a [16] STA05A	p29 PR-RS(1) RS-a [16] STA05A
SRSYNreq(r)				p25&p33 PR-RS(1) RS-r [16] STA05A	p25&p26&p33 PR-RS(1) RS-r [16] STA05A
SRSYNreq(s)				p25 PR-RS(1) RS-s [16] STA05A	p25&p26 PR-RS(1) RS-s [16] STA05A
SRSYNrsp					

TABLE A-11/X.225

Activity interrupt and discard state table

STATE / EVENT	STA01A await AA	STA01C idle TC con	STA01D await CDO	STA04A await PR or MAA	STA04B await PR or AEA	STA05A await PR or RA	STA05B await PR or AIA
AD	STA01A	TDISreq STA01	TDISreq STA01			p38&p40 SACTDind [16] STA11C	
ADA	STA01A	TDISreq STA01	TDISreq STA01				
AI	STA01A	TDISreq STA01	TDISreq STA01			p38&p40 SACTIind [16] STA11B	
AIA	STA01A	TDISreq STA01	TDISreq STA01				p38 SACTIcnf [29] STA713
SACTDreq				p34&p39 PR-RS(1) AD [16] STA05C	p39 PR-RS(1) AD [16] STA05C		
SACTDrsp							
SACTIreq				p34&p39 PR-RS(1) AI [16] STA05B	p39 PR-RS(1) AI [16] STA05B		
SACTIrsp							

TABLE A-11/X.225 *(cont.)*

Activity interrupt and discard state table

STATE EVENT	STA05C await PR or ADA	STA06 await RS after coll	STA10A await SSYNMrsp	STA10B await SACTErsp	STA11A await SRSYNrsp	STA11B await SACTIrsp
AD		p37&p40 SACTDind [16] STA11C	p38&p40 SACTDind [16] STA11C	p38&p40 SACTDind [16] STA11C		
ADA	p38 SACTDcnf [29] STA713					
AI		p37&p40 SACTIind [16] STA11B	p38&p40 SACTIind [16] STA11B	p38&p40 SACTIind [16] STA11B		
AIA						
SACTDreq					p34&p39 PR-RS(1) AD [16] STA05C	
SACTDrsp						
SACTIreq					p34&p39 PR-RS(1) AI [16] STA05B	
SACTIrsp						PR-RA(1) AIA [30] STA713

TABLE A-11/X.225 *(cont.)*

Activity interrupt and discard state table

STATE EVENT	STA11C await SACTDrsp	STA15A wait after PR-MAA	STA15B wait after PR-RS	STA15C wait after PR-RA	STA15D wait after PR-AB	STA16 await TDISind
AD			p37&p40 SACTDind [16] STA11C			STA16
ADA				p37&p41 SACTDcnf [29] STA713 [6]	STA15D	STA16
AI			p37&p40 SACTIind [16] STA11B		STA15D	STA16
AIA				p37&p42 SACTIcnf [29] STA713 [6]	STA15D	STA16
SACTDreq		p34&p39 PR-RS(1) AD [16] STA05C [6]	p27&p34&p39 PR-RS(1) AD [16] STA05C		STA15D	
SACTDrsp	PR-RA(1) ADA [30] STA713				STA15D	
SACTIreq		p34&p39 PR-RS(1) AI [16] STA05B [6]	p27&p34&p39 PR-RS(1) AI [16] STA05B		STA15D	
SACTIrsp					STA15D	

TABLE A-11/X.225 *(end)*

Activity interrupt and discard state table

STATE EVENT	STA19 await recovery (init)	STA20 await recovery	STA713 data transfer
AD	p38&p40 SACTDind [16] STA11C	p38&p40 SACTDind [16] STA11C	p38&p40 SACTDind [16] STA11C
ADA			
AI	p38&p40 SACTIind [16] STA11B	p38&p40 SACTIind [16] STA11B	p38&p40 SACTIind [16] STA11B
AIA			
SACTDreq		p34&p11 PR-RS(1) AD [16] STA05C	p34&p39 PR-RS(1) AD [16] STA05C
SACTDrsp			
SACTIreq		p34&p11 PR-RS(1) AI [16] STA05B	p34&p39 PR-RS(1) AI [16] STA05B
SACTIrsp			

TABLE A-12/X.225

Activity start, resume and capability data state table

STATE EVENT	STA01A await AA	STA01C idle TC con	STA01A await CDO	STA15B wait after PR-RS	STA15B wait after PR-AB	STA16 await TDISind	STA21 await CDA	STA22 await SCDrsp	STA713 data transfer
AR	STA01A	TDISreq STA01	TDISreq STA01	p44 SACTRind [12] [27] STA15B	STA15D	STA16			p44 SACTRind [12] [27] STA713 [6]
AS	STA01A	TDISreq STA01	TDISreq STA01	p44 SACTSind [12] [26] STA15B	STA15D	STA16			p44 SACTSind [12] [26] STA713 [6]
CD	STA01A	TDISreq STA01	TDISreq STA01		STA15D	STA16			p46 SCDind STA22
CDA	STA01A	TDISreq STA01	TDISreq STA01		STA15D	STA16	SCDcnf STA713		
SACTRreq					STA15D				p45 AR [12] [27] STA713
SACTSreq					STA15D				p45 AS [12] [26] STA713
SCDreq					STA15D				p47 CD STA21
SCDrsp					STA15D			CDA STA713	

TABLE A-13/X.225

Token management and exceptions state table

STATE EVENT	STA01A await AA	STA01C idle TC con	STA01D await CDO	STA03 await DN	STA04A await PR or MAA	STA04B await PR or AEA	STA05A await PR or RA	STA05B await PR or AIA
ED	STA01A	TDISreq STA01	TDISreq STA01	p52 SUERind STA20	p48&p03 SUERind STA20 p48&^p03 SUERind STA713	p48&p03 SUERind STA20 p48&^p03 SUERind STA713	p48 STA05A	p48 STA05B
ER	STA01A	TDISreq STA01	TDISreq STA01	p52 SPERind STA20	p48&p03 SPERind STA20 p48&^p03 SPERind STA713	p48&p03 SPERind STA20 p48&^p03 SPERind STA713	p48 STA05A	p48 STA05B
GT	STA01A	TDISreq STA01	TDISreq STA01		p59 SGTind [11] STA04A	p59 SGTind [11] STA04B	p59 STA05A	p59 STA05B
GTA	STA01A	TDISreq STA01	TDISreq STA01					
GTC	STA01A	TDISreq STA01	TDISreq STA01					
PT	STA01A	TDISreq STA01	TDISreq STA01	p53 SPTind STA03	p53 SPTind STA04A	p53 SPTind STA04B	p53 STA05A	p53 STA05B
SCGreq								
SGTreq					p54 GT [11] STA04A	p54 GT [11] STA04B		
SPTreq								
SUERreq								

TABLE A-13/X.225 *(cont.)*

Token management and exceptions state table

STATE EVENT	STA05C await PR or ADA	STA06 await RS after coll	STA09 await SRELrsp	STA10A await SSYNMrsp	STA10B await SACTErsp	STA15A wait after PR-MAA
ED	p48 STA05C	p48 STA06				
ER	p48 STA05C	p48 STA06				
GT	p59 STA05C	p59 STA06		p59 SGTind [11] STA10A	p59 SGTind [11] STA10B	p59 SGTind [11] STA15A
GTA						
GTC						
PT	p53 STA05C	p53 STA06				p53 SPTind STA15A
SCGreq						
SGTreq				p54 GT [11] STA10A	p54 GT [11] STA10B	p54 GT [11] STA15A
SPTreq			p53 PT STA09	p53 PT STA10A	p53 PT STA10B	
SUERreq			p50 ED STA19	p50 ED STA19	p50 ED STA19	

TABLE A-13/X.225 *(cont.)*

Token management and exceptions state table

STATE / EVENT	STA15B wait after PR-RS	STA15C wait after PR-RA	STA15D wait after PR-AB	STA16 await TDISind	STA18 await GTA	STA19 await recovery (init)
ED		p48 STA15C	STA15D	STA16		p50 SUERind STA19
ER		p48 STA15C	STA15D	STA16		p50 SPERind STA19
GT	p59 STA15B	p59 STA15C	STA15D	STA16		p60 SGTind [11] STA19 p61 SGTind [11] STA713
GTA			STA15D	STA16	STA713 [6]	
GTC			STA15D	STA16		
PT	p53 STA15B	p53 STA15C	STA15D	STA16	p53 SPTind STA18	p53 STA19
SCGreq			STA15D			
SGTreq	p54 STA15B		STA15D			
SPTreq	p53 STA15B		STA15D			
SUERreq	p50 STA15B		STA15D			

TABLE A-13/X.225 *(end)*

Token management and exceptions state table

STATE / EVENT	STA20 await recovery	STA21 await CDA	STA22 await SCDrsp	STA713 data transfer
ED				p50 SUERind STA713 p51 SUERind STA20
ER		p48 SPERind STA20		p50 SPERind STA713 p51 SPERind STA20
GT	p60 SGTind [11] STA20 p61 SGTind [11] STA713	p59 SGTind [11] STA21		p59 SGTind [11] STA713
GTA				
GTC				p62 SCGind GTA [11] STA713
PT	p53 STA20	p53 SPTind STA21		p53 SPTind STA713
SCGreq				p55 GTC [11] STA18
SGTreq	p57 GT [11] STA20 p58 GT [11] STA713			p54 GT [11] STA713
SPTreq			p53 PT STA22	p53 PT STA713
SUERreq				p50 ED STA19

TABLE A-14/X.225

Connection release state table

STATE / EVENT	STA01A await AA	STA01C idle TC con	STA01D await CDO	STA03 await DN	STA05A await PR or RA	STA06 await RS after coll	STA09 await SRELrsp
DN	STA01A	TDISreq STA01	TDISreq STA01	îp66 SRELcnf+ TDISreq STA01 p66 SRELcnf+ STA01C			p69&îp01 SRELcnf+ [32] STA09
FN-nr	STA01A	TDISreq STA01	TDISreq STA01	ˆp65 SRELind [8] [18] STA09	p68 STA05A	p68 STA06	
FN-r	STA01A	TDISreq STA01	TDISreq STA01	ˆp65&ˆp01&p16 SRELind [8] [18] STA09	p68&ˆp01&p16 STA05A		
NF	STA01A	TDISreq STA01	TDISreq STA01	p67 SRELcnf− STA713			
SRELreq							p65 FN-nr [8] [18] STA09
SRELrsp+							ˆp66&p75 DN [4] STA16 p66 DN STA01C p69&p01 DN STA03
SRELrsp−							p67 NF STA713

TABLE A-14/X.225 *(end)*

Connection release state table

STATE EVENT	STA15B wait after PR-RS	STA15C wait after PR-RA	STA15D wait after PR-AB	STA16 await TDISind	STA19 await recovery (init)	STA20 await recovery	STA713 data transfer
DN				STA16			
FN-nr		p68 STA15C	STA15D	STA16	p68 STA19	p68 STA20	p68 SRELind [8] STA09
FN-r		p68&^p01&p16 STA15C		STA16	p68&^p01&p16 STA19	p68&^p01&p16 STA20	p68&^p01&p16 SRELind [9] STA09
NF	p67 SRELcnf− STA15B		STA15D	STA16			
SRELreq	p63 STA15B		STA15D				p63&^p64 FN-nr [8] STA03 p63&p64 FN-r [7] STA03
SRELrsp+			STA15D				
SRELrsp−			STA15D				

TABLE A-15/X.225

Abort state table

STATE EVENT	STA01 idle no TC	STA01A await AA	STA01B await TCONcnf	STA01C idle TC con	STA01D await CDO	STA02A await AC	STA02B await OA	STA03 await DN	STA04A await PR or MAA
AA	/ /	[3] STA01C	/ /	TDISreq STA01	TDISreq STA01				
AB-nr	/ /	[3] TDISreq STA01	/ /	TDISreq STA01	TDISreq STA01	SxABind(3) TDISreq STA01	SxABind(3) TDISreq STA01	SxABind(3) TDISreq STA01	SxABind(3) TDISreq STA01
AB-r	/ /	[3] STA01C	/ /	ˆp02 TDISreq STA01 p02 AA STA01C	ˆp02 TDISreq STA01 p02 AA STA01C	ˆp02 SxABind(3) TDISreq STA01 p02 SxABind(3) AA STA01C	ˆp02 SxABind(3) TDISreq STA01 p02 SxABind(3) AA STA01C	ˆp02 SxABind(3) TDISreq STA01 p02 SxABind(3) AA STA01C	ˆp02 SxABind(3) TDISreq STA01 p02 SxABind(3) AA STA01C
PR-AB	/ /	/ /	/ /	TDISreq STA01	/ /	STA15D	/ /	STA15D	STA15D
SUABreq			TDISreq STA01			ˆp02 PR-AB(4) AB-nr [4] STA16 p02 AB-r [4] STA01A	ˆp02 PR-AB(4) AB-nr [4] STA16 p02 AB-r [4] STA01A	ˆp02 PR-AB(4) AB-nr [4] STA16 p02 AB-r [4] STA01A	ˆp02 PR-AB(4) AB-nr [4] STA16 p02 AB-r [4] STA01A
TDISind	/ /	[3] STA01	SPABind STA01	STA01	STA01	SPABind STA01	SPABind STA01	SPABind STA01	SPABind STA01
TIM	/ /	TDISreq STA01	/ /	/ /	/ /	/ /	/ /	/ /	/ /

TABLE A-15/X.225 *(cont.)*

Abort state table

STATE EVENT	STA04B await PR or AEA	STA05A await PR or RA	STA05B await PR or AIA	STA05C await PR or ADA	STA06 await RS after coll	STA08 await SCONrsp
AA						
AB-nr	SxABind(3) TDISreq STA01	SxABind(3) TDISreq STA01	SxABind(3) TDISreq STA01	SxABind(3) TDISreq STA01	SxABind(3) TDISreq STA01	SxABind(3) TDISreq STA01
AB-r	ˆp02 SxABind(3) TDISreq STA01 p02 SxABind(3) AA STA01C	ˆp02 SxABind(3) TDISreq STA01 p02 SxABind(3) AA STA01C	ˆp02 SxABind(3) TDISreq STA01 p02 SxABind(3) AA STA01C	ˆp02 SxABind(3) TDISreq STA01 p02 SxABind(3) AA STA01C		ˆp02 SxABind(3) TDISreq STA01 p02 SxABind(3) AA STA01C
PR-AB	STA15D	STA15D	STA15D	STA15D	STA15D	STA15D
SUABreq	ˆp02 PR-AB(4) AB-nr [4] STA16 p02 AB-r [4] STA01A	ˆp02 PR-AB(4) AB-nr [4] STA16 p02 AB-r [4] STA01A	ˆp02 PR-AB(4) AB-nr [4] STA16 p02 AB-r [4] STA01A	ˆp02 PR-AB(4) AB-nr [4] STA16 p02 AB-r [4] STA01A	ˆp02 PR-AB(4) AB-nr [4] STA16	ˆp02 PR-AB(4) AB-nr [4] STA16 p02 AB-r [4] STA01A
TDISind	SPABind STA01	SPABind STA01	SPABind STA01	SPABind STA01	SPABind STA01	SPABind STA01
TIM	/ /	/ /	/ /	/ /	/ /	/ /

TABLE A-15/X.225 *(cont.)*

Abort state table

STATE EVENT	STA09 await SRELrsp	STA10A await SSYNMrsp	STA10B await SACTErsp	STA11A await SRSYNrsp	STA11B await SACTIrsp	STA11C await SACTDrsp
AA						
AB-nr	SxABind(3) TDISreq STA01	SxABind(3) TDISreq STA01	SxABind(3) TDISreq STA01	SxABind(3) TDISreq STA01	SxABind(3) TDISreq STA01	SxABind(3) TDISreq STA01
AB-r	ˆp02 SxABind(3) TDISreq STA01 p02 SxABind(3) AA STA01C	ˆp02 SxABind(3) TDISreq STA01 p02 SxABind(3) AA STA01C	ˆp02 SxABind(3) TDISreq STA01 p02 SxABind(3) AA STA01C	ˆp02 SxABind(3) TDISreq STA01 p02 SxABind(3) AA STA01C	ˆp02 SxABind(3) TDISreq STA01 p02 SxABind(3) AA STA01C	ˆp02 SxABind(3) TDISreq STA01 p02 SxABind(3) AA STA01C
PR-AB	STA15D	STA15D	STA15D	STA15D	STA15D	STA15D
SUABreq	ˆp02 PR-AB(4) AB-nr [4] STA16 p02 AB-r [4] STA01A	ˆp02 PR-AB(4) AB-nr [4] STA16 p02 AB-r [4] STA01A	ˆp02 PR-AB(4) AB-nr [4] STA16 p02 AB-r [4] STA01A	ˆp02 PR-AB(4) AB-nr [4] STA16 p02 AB-r [4] STA01A	ˆp02 PR-AB(4) AB-nr [4] STA16 p02 AB-r [4] STA01A	ˆp02 PR-AB(4) AB-nr [4] STA16 p02 AB-r [4] STA01A
TDISind	SPABind STA01	SPABind STA01	SPABind STA01	SPABind STA01	SPABind STA01	SPABind STA01
TIM	/ /	/ /	/ /	/ /	/ /	/ /

TABLE A-15/X.225 *(cont.)*

Abort state table

STATE EVENT	STA15A wait after PR-MAA	STA15B wait after PR-RS	STA15C wait after PR-RA	STA15D wait after PR-AB	STA16 await TDISind	STA18 await GTA	STA19 await recovery (init)
AA					[3] TDISreq STA01		
AB-nr	SxABind(3) TDISreq STA01	SxABind(3) TDISreq STA01	SxABind(3) TDISreq STA01	SxABind(3) TDISreq STA01	[3] TDISreq STA01	SxABind(3) TDISreq STA01	SxABind(3) TDISreq STA01
AB-r					[3] TDISreq STA01	ˆp02 SxABind(3) TDISreq STA01 p02 SxABind(3) AA STA01C	ˆp02 SxABind(3) TDISreq STA01 p02 SxABind(3) AA STA01C
PR-AB	STA15D	STA15D	STA15D		[3] TDISreq STA01	STA15D	STA15D
SUABreq	ˆp02 PR-AB(4) AB-nr [4] STA16	ˆp02 PR-AB(4) AB-nr [4] STA16	ˆp02 PR-AB(4) AB-nr [4] STA16	[4] STA15D		ˆp02 PR-AB(4) AB-nr [4] STA16 p02 AB-r [4] STA01A	ˆp02 PR-AB(4) AB-nr [4] STA16 p02 AB-r [4] STA01A
TDISind	SPABind STA01	SPABind STA01	SPABind STA01	SPABind STA01	[3] STA01	SPABind STA01	SPABind STA01
TIM	/ /	/ /	/ /	TDISreq STA01	TDISreq STA01	/ /	/ /

TABLE A-15/X.225 *(end)*

Abort state table

STATE EVENT	STA20 await recovery	STA21 await CDA	STA22 await SCDrsp	STA713 data transfer
AA				
AB-nr	SxABind(3) TDISreq STA01	SxABind(3) TDISreq STA01	SxABind(3) TDISreq STA01	SxABind(3) TDISreq STA01
AB-r	^p02 SxABind(3) TDISreq STA01 p02 SxABind(3) AA STA01C	^p02 SxABind(3) TDISreq STA01 p02 SxABind(3) AA STA01C	^p02 SxABind(3) TDISreq STA01 p02 SxABind(3) AA STA01C	^p02 SxABind(3) TDISreq STA01 p02 SxABind(3) AA STA01C
PR-AB	STA15D	STA15D	STA15D	STA15D
SUABreq	^p02 PR-AB(4) AB-nr [4] STA16 p02 AB-r [4] STA01A	^p02 PR-AB(4) AB-nr [4] STA16 p02 AB-r [4] STA01A	^p02 PR-AB(4) AB-nr [4] STA16 p02 AB-r [4] STA01A	^p02 PR-AB(4) AB-nr [4] STA16 p02 AB-r [4] STA01A
TDISind	SPABind STA01	SPABind STA01	SPABind STA01	SPABind STA01
TIM	/ /	/ /	/ /	/ /

ANNEX B

(to Recommendation X.225)

Relationship to CCITT Recommendation T.62 encoding

This Recommendation has been designed to be compatible with Recommendation T.62.

Table B-1/X.225 shows the relationship between the Recommendation T.62 commands and responses and the SPDUs used in this Recommendation.

Table B-2/X.225 shows the relationship between the Recommendation T.62 PGI and PI parameters and the PGI and PI parameters used in this Recommendation.

Annex C lists the PGIs and PIs which are not defined in this Recommendation, but are reserved as they are used in Recommendation T.62 for parameters relevant to higher layers than the session layer. Use of these PGIs and PIs is necessary for correct operation to Recommendation T.62. An implementation of the protocol specified in this Recommendation will need to be enhanced to take account of these PGI and PI units.

TABLE B-1/X.225

Relationship between T.62 commands and responses and X.225 SPDUs

Code	T.62 name	SPDU Code	SPDU name
13	CSS	CN	CONNECT
16	xxxx	OA	OVERFLOW ACCEPT
15	xxxx	CDO	CONNECT DATA OVERFLOW
14	RSSP	AC	ACCEPT
12	RSSN	RF	REFUSE
9	CSE	FN	FINISH
10	RSEP	DN	DISCONNECT
25	CSA	AB	ABORT
26	RSAP	AA	ABORT ACCEPT
1	CSUI-CDUI	DT	DATA TRANSFER
2	RSUI	PT	PLEASE TOKENS
21	CSCC	GTC	GIVE TOKENS CONFIRM
22	RSCCP	GTA	GIVE TOKENS ACK
1	CSUI	GT	GIVE TOKENS
0	RSUI-RDGR	ER	EXCEPTION REPORT
48	RSUI-RDPBN	ED	EXCEPTION DATA
33	CSTD	TD	TYPED DATA
8	xxxx	NF	NOT FINISHED
49	CSUI-CDPB	MIP	MINOR SYNC POINT
50	RSUI-RDPBP	MIA	MINOR SYNC ACK
41	CSUI-CDE	MAP	MAJOR SYNC POINT
42	RSUI-RDEP	MAA	MAJOR SYNC ACK
7	xxxx	PR	PREPARE
53	xxxx	RS	RESYNCHRONIZE
34	xxxx	RA	RESYNCHRONIZE ACK
5	xxxx	EX	EXPEDITED DATA
45	CSUI-CDS	AS	ACTIVITY START
29	CSUI-CDC	AR	ACTIVITY RESUME
25	CSUI-CDR	AI	ACTIVITY INTERRUPT
26	RSUI-RDRP	AIA	ACTIVITY INTERRUPT ACK
57	CSUI-CDD	AD	ACTIVITY DISCARD
58	RSUI-RDDP	ADA	ACTIVITY DISCARD ACK
41	CSUI-CDE	AE	ACTIVITY END
42	RSUI-RDEP	AEA	ACTIVITY END ACK
61	CSUI-CDCL	CD	CAPABILITY DATA
62	RSUI-RDCLP	CDA	CAPABILITY DATA ACK

TABLE B-2/X.225

Relationship between Recommendation T.62 PGI/PI and Recommendation X.225 parameters

T.62 parameters	code	X.225 parameters
PGI		
Reserved for extension	0	See Table C-1/X.225
Session reference	1	Connection Identifier
Non-basic session capabilities	2	See Table C-1/X.225
	3	
	4	
	5	Connect/Accept Item
	6	
	7	
PI		
Service identifier	8	See Table C-1/X.225
Terminal identifier (called terminal)	9	Called SS-user Ref.
Terminal identifier (calling terminal)	10	Calling SS-user Ref.
Date and time	11	Common Reference
Additional session reference number	12	Additional Ref. Info.
Miscellaneous session capabilities	13	See Table C-1/X.225
Window size	14	See Table C-1/X.225
	15	Sync Type Item
Session control functions	16	Token Item
Session termination parameter	17	Transport Disconnect
Inactivity timer	18	See Table C-1/X.225
	19	Protocol options
Session service functions	20	Session Requirements
	21	TSDU Maximum Size
	22	Version number
	23	Initial Serial Number
	24	Prepare Type
	25	Enclosure Item
	26	Token Setting Item
	27	Resync Type
Initiator's reference number	28	See Table C-1/X.225
Acceptor's reference number	29	See Table C-1/X.225
Reactivation/transaction indication	30	See Table C-1/X.225
Suspend reject reason	31	See Table C-1/X.225
PGI		
Reserved for extension	32	See Table C-1/X.225
Document linking	33	Linking Information
	34	
	35	
	36	
	37	
	38	
	39	

TABLE B-2/X.225 *(cont.)*

T.62 parameters	code	X.225 parameters
PI		
Service interworking identifier	40	See Table C-1/X.225
Document reference number	41	Activity identifier
Checkpoint reference number	42	Serial number
Reserved	43	
Acceptance of CDCL parameters	44	See Table C-1/X.225
Storage capacity negotiation	45	See Table C-1/X.225
Receiving ability jeopardized	46	User data (in MIA SPDU)
Reserved	47	
Document type identifier	48	See Table C-1/X.225
Reflect parameter values	49	Reflect parameter values
Reason (session and document)	50	Reason code
	51	Calling session selector
	52	Called/Responding session selector
	53	
	54	
	55	
	56	
	57	
	58	
	59	
	60	Data overflow
	61	
	62	
	63	
PGI		
Reserved for extension	64	See Table C-1/X.225
Nonbasic teletex terminal capabilities	65	See Table C-1/X.225
	66	
	67	
	68	
	69	
	70	
	71	
PI		
Graphic character set	72	See Table C-1/X.225
Control character set	73	See Table C-1/X.225
Teletex page format	74	See Table C-1/X.225
Miscellaneous teletex terminal capabilities	75	See Table C-1/X.225
	76	
Number of dots for character box height	77	See Table C-1/X.225
Number of dots for character box width	78	See Table C-1/X.225
	79	
PGI		
	192	
Session user data	193	User data
	194	Extended User Data

ANNEX C

(to Recommendation X.225)

PGIs and PIs reserved for use by Recommendation T.62

Table C-1/X.225 lists the PGIs and PIs which are not defined in this Recommendation, but which are reserved as they are used in Recommendation T.62 for parameters that are relevant to higher layers than the session layer.

TABLE C-1/X.225

PGIs and PIs reserved for use by Recommendation T.62

PGI	
0	Reserved for extension
1	Non-basic session capabilities
PI	
8	Service identifier
13	Miscellaneous session capabilities
14	Window size
18	Inactivity timer
28	Initiator's reference number
29	Acceptor's reference number
30	Reactivation/transaction indication
31	Suspend reject reason
PGI	
32	Reserved for extension
PI	
40	Service interworking identifier
44	Acceptance of CDCL parameters
45	Storage capacity negotiation
48	Document type identifier
PGI	
64	Reserved for extension
65	Non-basic teletex terminal capabilities
PI	
72	Graphic character set
73	Control character set
74	Teletex page format
75	Miscellaneous teletex terminal capabilities
77	Number of dots for character box height
78	Number of dots for character box width

ANNEX D

(to Recommendation X.225)

Compatibility between protocol version 1 and protocol version 2

Protocol version 2 of the Session Protocol is a superset of protocol version 1 (both of which are specified in this Recommendation). Protocol version 1 of the Session Protocol imposes restrictions on the length of user data fields. Protocol version 2 removes those length restrictions.

An implementation of the session protocol may limit the length of user data supported, based on the requirements of its SS-user or protocol version supported. Any such limitation is stated in the Protocol Implementation Conformance Statement. If no user of a session implementation requires more than 10k of user data during connection establishment, the implementation need not be able to send the CDO SPDU or receive the OA SPDU.

Implementations of protocol version 2 can interwork with implementations of protocol version 1 only by imposing a number of restrictions (all of which are valid, in terms of the conformance statement). These restrictions are:

a) User Data parameter value in the ABORT SPDU shall not exceed 9 octets.

b) Reason Code parameter value in the REFUSE SPDU shall not exceed 513 octets.

c) The User Data PGI unit shall not be present in GIVE TOKENS, GIVE TOKENS CONFIRM, ACTIVITY INTERRUPT, ACTIVITY INTERRUPT ACK, ACTIVITY DISCARD and ACTIVITY DISCARD ACK SPDUs. The User Data PGI unit in all other SPDUs shall not exceed 512 octets.

d) Protocol version 1 shall be proposed in the CONNECT SPDU. In this case the Extended User Data parameter and Data Overflow parameter in the CONNECT SPDU shall not be present.

Note – Protocol version 2 may also be proposed, but to operate validly with a protocol version 1 only implementation, protocol version 1 shall be selected.

As a consequence of protocol version 1 being selected:

e) segmenting, as specified in § 6.3.5 b) does not apply. Only data SSDUs and typed data SSDUs may be segmented;

f) the OVERFLOW ACCEPT SPDU and CONNECT DATA OVERFLOW SPDU are not used.

Note – Implementations of the earlier edition of this Recommendation which specified protocol version 1 can upgrade to and claim conformance with this edition of this Recommendation by stating, in their Protocol Implementation Conformance Statement, the restrictions specified in a) to c) above; and by conforming to the specified procedure for rejecting SPDUs with "to much" user data (see § A.4.3.1.2) (note that this requires that the implemantation recognises the Extended User Data parameter in the CONNECT SPDU and the Enclosure Item in the ABORT SPDU). This is a minimal implementation of protocol version 2 and will not satisfy the requirements of some ASEs.

APPENDIX I

(to Recommendation X.225)

Differences between Recommendation X.225 and ISO International Standard 8327

I.1 In ISO 8327 the last sentence in the second paragraph of A.1 states "In case of arbitration or dispute this annex takes precedence over clause 7". This statement is not included in this Recommendation.

Recommendation X.226

PRESENTATION PROTOCOL SPECIFICATION FOR OPEN SYSTEMS INTERCONNECTION FOR CCITT APPLICATIONS[1]

(Melbourne, 1988)

The CCITT,

considering

(a) that Recommendation X.200 defines the Reference Model of Open System Interconnection for CCITT Applications;

(b) that Recommendation X.208 specifies Abstract Syntax Notation One (ASN.1) for the specification of the Abstract Syntax of Protocols;

(c) that Recommendation X.209 specifies the Basic Encoding Rules for Abstract Syntax Notation One;

(d) that Recommendation X.210 defines the Open Syntax Interconnection (OSI) Layer Service Definition Conventions;

(e) that Recommendation X.215 defines the Session Service Definition for Open Systems Interconnection for CCITT Applications;

(f) that Recommendation X.216 defines the Presentation Service Definition for Open Systems Interconnection for CCITT Applications;

(g) that Recommendation X.410-1984 specifies the protocol for Remote Operation and Reliable Transfer Server for Message Handling Systems,

unanimously declares

that this Recommendation defines the Presentation Protocol of Open Systems Interconnection for CCITT Applications as given in the Scope and Field of Application.

CONTENTS

[1] Recommendation X.226 and ISO 8823 [Information processing systems – Open Systems Intrconnection – Connection oriented presentation protocol specification] were developed in close collaboration and are technically aligned, except for the differences noted in Appendix I

0 Introduction

This Recommendation is one of a set of Recommendations produced to facilitate the interconnection of information processing systems. It is related to other Recommendations in the set as defined by the Reference Model for Open Systems Interconnection (Recommendation X.200). The Reference Model subdivides the area of standardization for interconnection into a series of layers of specification, each of manageable size.

This Recommendation specifies a common encoding and a number of functional units of presentation protocol procedures to be used to meet the needs of presentation-service-users. It is intended that the presentation protocol should be simple but general enough to cater for the total range of presentation-service-user needs without restricting future extensions.

The primary aim of this Recommendation is to provide a set of rules for communication expressed in terms of the procedures to be carried out by peer entities at the time of communication. These rules for communication are intended to provide a sound basis for development in order to serve a variety of purposes:

a) as a guide for implementors and designers;

b) for use in the testing and procurement of equipment;

c) as part of an agreement for the admittance of systems into the open systems environment;

d) as a refinement of the understanding of OSI.

It is expected that the inifial users of this Recommendation will be designers and implementors of equipment and therefore it contains, in notes or in annexes, guidance on the implementation of its procedures.

It has not been possible as yet to prepare a product standard containing a set of objective tests for conformance to this Recommendation but it does contain a section on conformance of equipment claiming to implement the procedures it specifies. Attention is drawn to the fact that this Recommendation does not contain any test to demonstrate this conformance and cannot, therefore, be considered as a complete product standard.

The variations and options available within this Recommendation are essential to enable a presentation-service to be provided for a wide variety of applications. Thus, a minimally conforming implementation will not be suitable for use in all possible circumstances. It is necessary, therefore, to qualify all references to this Recommendation with statements of the options provided or required, or with statements of the intended purpose of provision or use.

1 Scope and field of application[2)]

1.1 This Recommendation specifies:

a) procedures for the transfer of data and control information from one presentation-entity to a peer presentation-entity;

b) the means of selecting, by means of functional units, the procedures to be used by the presentation-entities;

c) the structure and encoding of the presentation-protocol-data-units used for the transfer of data and control information.

The procedures are defined in terms of:

d) the interactions between peer presentation-entities through the exchange of presentation-protocol-data-units;

e) the interactions between a presentation-entity and the presentation-service-user in the same system through the exchange of presentation-service primitives;

f) the interactions between a presentation-entity and the session-service-provider through the exchange of session-service primitives.

[2)] The implementation and use of this Recommendation for Open Systems Interconnection requires the public assignment of values of ASN.1 type OBJECT IDENTIFIER to specifications of abstract syntaxes and transfer syntaxes. Public specification and naming of abstract syntaxes and transfer syntaxes can occur in ISO standards or CCITT Recommendations, or under the mechanisms identified in the Registration Authority procedures. A Registration Authority procedures specification is under development.

1.2 These procedures are defined in the main text of this Recommendation supplemented by state tables in annex A.

1.3 These procedures are applicable to instances of communication between systems which support the Presentation Layer of the OSI Reference Model and which wish to interconnect in an OSI environment.

1.4 This Recommendation also specifies conformance criteria for systems implementing these procedures. It does not contain tests which can be used to demonstrate this conformance.

2 References

Recommendation X.200 – Reference Model of Open Systems Interconnection for CCITT Applications. (See also ISO 7498).

Recommendation X.210 – OSI Layer Service Definition Conventions (see also ISO TR 8509).

ISO 7498-3 – Information processing systems – Open Systems Interconnection – basic Reference Model – Part 3: Naming and Addressing[3].

Recommendation X.215 – Session Service Definition for Open Systems Interconnection for CCITT Applications (see also ISO 8326 and ISO 8326 addendum 2).

Recommendation X.208 – Specification of Abstract Syntax Notation One (ASN.1) for CCITT Applications (see also ISO 8824).

Recommendation X.209 – Specification of Basic Encoding Rules for Abstract Syntax Notation One. (See also ISO 8825).

Recommendation X.216 – Presentation Service Definition for Open Systems Interconnection for CCITT Applications (see also ISO 8822).

Recommendation X.410-1984 – 1984 Message Handling Systems: Remote Operation and Reliable Transfer Server.

3 Definitions

3.1 *Reference Model definitions*

This Recommendation is based on the concepts developed in Recommendation X.200 and makes use of the following terms derived from it:

a) presentation-connection;

b) Presentation Layer;

c) presentation-protocol-data-unit;

d) presentation-service;

e) presentation-service-access-point;

f) presentation-service-data-unit;

g) presentation-protocol-control-information;

h) session-connection;

i) Session Layer;

j) session-service-acces-point;

k) session-service-data-unit;

l) session-service-provider;

m) transfer syntax.

3.2 *Service conventions definitions*

This Recommendation makes use of the following terms defined in Recommendation X.210 as they apply in the Presentation Layer:

a) service-user;

b) service-provider;

c) service primitive;

d) request;

[3] At present at the stage of draft.

e) indication;
f) response;
g) confirm;
h) non-confirmed-service;
i) confirmed-service;
j) provider-initiated-service.

3.3 *Naming and Addressing definitions*

This Recommendation makes use of the following terms defined in ISO 7498-3:

a) session-address;
b) presentation-address;
c) presentation-selector.

3.4 *Presentation Service definitions*

This Recommendation is also based on concepts developed in Recommendation X.216 and makes use of the following terms defined in that Recommendation:

a) abstract syntax;
b) abstract syntax name;
c) transfer syntax name;
d) presentation data value;
e) presentation context;
f) defined context set;
g) inter-activity defined context set;
h) default conext;
i) functional unit;
j) X.410-1984 mode;
k) normal mode.

3.5 *Presentation protocol definitions*

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

3.5.1 **local matter**

A decision made by a system concerning its behaviour in the Presentation Layer that is not subject to the requirements of this Recommendation.

3.5.2 **valid presentation-protocol-data-unit**

A presentation-protocol-data-unit which complies with the requirements of this Recommendation for structure and encoding.

3.5.3 **invalid presentation-protocol-data-unit**

A presentation-protocol-data-unit which does not comply with the requirements of this Recommendation for structure and encoding.

3.5.4 **protocol error**

A situation occuring when a presentation-protocol-data-unit is used in a way which does not comply with the procedures defined in this Recommendation.

3.5.5 **original activity identifier**

An attribute of an activity in progress. If the activity was started by use of the P-ACTIVITY-START service, the Activity identifier parameter value of the request and indication service primitives; if the activity was resumed by use of the P-ACTIVITY-RESUME service the Old activity identifier parameter value of the request and indication service primitives.

3.5.6 self-delimiting

An attribute of a transfer syntax which indicates that the end of each value in that syntax can be determined by means provided by the syntax.

3.5.7 presentation context identifier

An identifier for a specific presentation context. The identifier is unique within a presentation-connection and known to both presentation protocol machines. The default context does not have a presentation context identifier associated with it.

3.5.8 syncpoint identifier

A synchronization point serial number if the session activity management functional unit has not been selected; or a pair of synchronization point serial number and original activity identifier of the activity in progress if the session activity management functional unit has been selected. The order of syncpoint identifiers is defined as the order of their synchronization point serial number components.

3.5.9 initiator

The presentation protocol machine that initiates the presentation-connection establishment.

3.5.10 responder

The presentation protocol machine that responds to a presentation-connection establishment proposal.

3.5.11 requestor

The presentation protocol machine that initiates a particular action.

3.5.12 acceptor

The presentation protocol machine that accepts a particular action.

4 Abbreviations

4.1 *Data Units*

PPDU	presentation-protocol-data-unit
PSDU	presentation-service-data-unit
SSDU	session-service-data-unit

4.2 *Types of presentation-protocol-data-units*

AC PPDU	Alter Context PPDU
ACA PPDU	Alter Context Acknowledge PPDU
ARP PPDU	Abnormal Release Provider PPDU
ARU PPDU	Abnormal Release User PPDU
CP PPDU	Connect Presentation PPDU
CPA PPDU	Connect Presentation Accept PPDU
CPR PPDU	Connect Presentation Reject PPDU
RS PPDU	Resynchronize PPDU

RSA PPDU	Resynchronize Acknowledge PPDU
TC PPDU	Capacility Data PPDU
TCC PPDU	Capability Data Acknowledge PPDU
TD PPDU	Presentation Data PPDU
TE PPDU	Expedited Data PPDU
TTD PPDU	Presentation typed Data PPDU

4.3 *Other abbreviations*

ASN.1	Abstract Syntax Notation One (see Recommendation X.208)
DCS	defined context set
PPCI	presentation-protocol-control-information
PPM	presentation protocol machine
PS	presentation-service
PSAP	presentation-service-access-point
PS-user	presentation-service-user
SS	session-service
SSAP	session-service-access-point

5 **Overview of the presentation protocol**

5.1 *Service provided by the Presentation Layer*

The protocol specified in this Recommendation supports the presentation-service defined in Recommendation X.216.

5.2 *Service assumed from the Session Layer*

The protocol specified in this Recommendation assumes the use of the session-service defined in Recommendation X.215.

5.3 *Functions of the Presentation Layer*

The functions of the Presentation Layer are described in the Reference Model, Recommendation X.200, and are further expanded in the Presentation Service Definition, Recommendation X.216.

5.4 *Presentation functional units*

Functional units are logical groupings of elements of procedure defined by this Recommendation for the purpose of:

a) negotiation during presentation-connection establishment for subsequent use on the presentation-connection;

b) specification of conformance requirements.

The selection of the presentation functional units does not constrain the selection of session functional units to be available to the PS-user. Selection of a particular session functional unit to be available to the PS-user implies the rules of interaction of that session functional unit with whatever presentation functional units are selected, as pecified by this Recommendation.

5.4.1 *Kernel functional unit*

This functional unit, which is always available, supports the basic protocol elements of procedure required to establish a presentation-connection, transfer data, and release the presentation-connection.

Note – This is the presentation kernel functional unit; it supports data transfer on whatever session functional units are selected for those presentation-service primitives which allow User data parameters.

5.4.2 *Context management functional unit*

This functional unit supports the context addition and deletion services. This functional unit is optional, and its use is negotiable.

5.4.3 *Context restoration functional unit*

This functional unit adds further Presentation Layer functions when the session activity management functional unit is selected or when both the session synchronization (major or minor) and the session resynchronization functional units are selected. The context restoration functional unit is optional, and its use is negotiable; it is available only when the context management functional is selected.

5.5 *Model of the Presentation Layer*

The presentation protocol machine (PPM), see the note, within the presentation-entity communicates with the PS-user through a PSAP by means of presentation-service primitives as defined by the Presentation Service Definition (Recommendation X.216). Presentation-service primitives will cause or be the result of presentation-protocol-data-unit (PPDU) exchanges between the peer PPMs using a session-connection. These protocol exchanges are effected using the services of the Session Layer as defined by the Session Service Definition (Recommendation X.215). In some cases, presentation-service primitives will directly cause or be the result of session-service primitives.

Presentation-connection-endpoints are identified in end systems by an internal, implementation dependent, mechanism so that the PS-user and the presentation-entity can refer to each presentation-connection.

The reception of a service primitive and the generation of dependent actions are considered to be an indivisible action. The reception of a PPDU and the generation of dependent actions are considered to be an indivisible action.

The model of the Presentation Layer for a single presentation-connection is illustrated in Figure 1/X.226.

Note – A presentation entity is comprised of one or more PPMs.

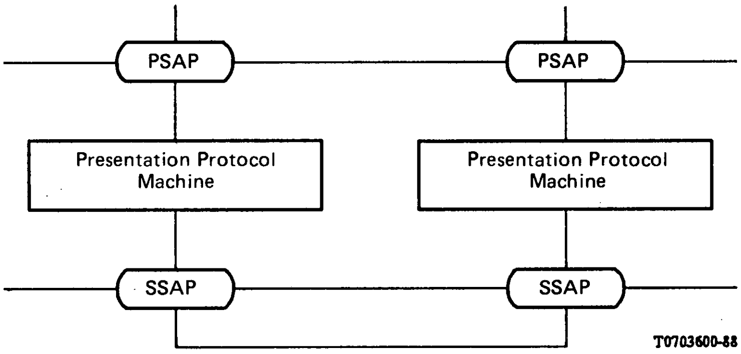


FIGURE 1/X.226

Model of the Presentation Layer

6 Elements of procedure

For the purpose of description, this specification of elements of procedure employs an integrated treatment of PPDU parameters and session-service primitive parameters. This section does not identity a parameter as either a PPDU parameter or a session-service primitive parameter. Such a distinction is specified in § 7. For further information on the use of parameters, refer to the Presentation Service Definition (Recommendation X.216).

6.1 *User data parameters*

Most of the PPDUs used in the procedures of the presentation protocol carry User data parameters containing one or more presentation data values. The remainder of the section gives the rules for determining the presentation contexts from which these presentation data values (including any embedded presentation data values) shall be taken.

Note – If the underlying session-service-provider imposes a restriction on the length of certain SS-user data parameters, the PPM shall reject any presentation-service request or response primitive (with the exception of a P-U-ABORT request primitive, see § 6.4.2.2) carrying a User data parameter which does not fit into the SS-user data parameter of the corresponding session-service primitive. The way in which the PPM is made aware of this is a local matter.

6.1.1 The presentation data values (including any embedded presentation data values) which may be transferred in the User data parameter of the TE PPDU shall always be from the default context.

6.1.2 The presentation data values (including any embedded presentation data values) in User data parameters except for the TE PPDU shall be from presentation contexts determined by the following rules:

a) If the DCS is empty and (d) does not apply, then each presentation data value (including any embedded presentation data values) shall be from the default context.

b) If the DCS is not empty and no procedure is in progress which can amend the contents of the DCS, then each presentation data value (including any embedded presentation data values) shall be from a presentation context of the DCS.

c) If the element of procedure itself amends the DCS, then each presentation data value (including any embedded presentation data values) shall be from a presentation context of the DCS which results from this amendment, or from the default context if this amendment leaves the DCS empty.

d) If a PPM is awaiting a PPDU which will confirm a proposed amendment to the DCS, then each presentation data value (including any embedded presentation data values) shall be from a presentation context of the DCS which was not proposed for deletion from the DCS. If this leaves no presentation contexts available, then the User data parameter shall not be present.

6.2 *Connection establishment*

6.2.1 *Purpose*

The connection establishment procedure is used to establish a presentation-connection between two presentation-entities. It is used by a PPM which has received a P-CONNECT request service primitive.

The procedure uses the following PPDUs:

a) CP PPDU;

b) CPA PPDU;

c) CPR PPDU.

6.2.2 *CP PPDU associated parameters*

6.2.2.1 *Mode selector*

This shall be the Mode parameter from the P-CONNECT request service primitive and shall identify the mode of operation of the PPM for this presentation-connection. It shall appear as the Mode parameter of the P-CONNECT indication service primitive, if issued.

6.2.2.2 *Protocol version*

This shall identify each version of the presentation protocol that the initiating PPM supports. The version of the protocol defined in this Recommendation shall be version-1.

See also § 6.2.6.4.

6.2.2.3 *Calling-presentation-selector*

This shall be the presentation-selector part of the Calling-presentation-address parameter from the P-CONNECT request service primitive and shall appear as the calling-presentation-selector part of the Calling-presentation-address parameter of the P-CONNECT indication service primitive, if issued.

6.2.2.4 *Calling-session-address*

This shall be the session-address part of the Calling-presentation-address parameter from the P-CONNECT request service primitive and shall appear as the session-address part of the Calling-presentation-address parameter of the P-CONNECT indication service primitive, if issued.

6.2.2.5 *Called-presentation-selector*

This shall be the presentation-selector part of the Called-presentation-address parameter from the P-CONNECT request service primitive and shall appear as the called-presentation-selector part of the Called-presentation-address parameter of the P-CONNECT indication service primitive, if issued.

6.2.2.6 *Called-session-address*

This shall be the session-address part of the Called-presentation-address parameter from the P-CONNECT request service primitive and shall appear as the session-address part of the Called-presentation-address parameter of the P-CONNECT indication service primitive, if issued.

6.2.2.7 *Presentation context definition list*

This shall be a list containing one or more items. Each item represents one item of the Presentation context definition list parameter from the P-CONNECT request service primitive and shall appear as one item of the Presentation context definition list parameter of the P-CONNECT indication service primitive, if issued. Each item contains three components: a presentation context identifier, an abstract syntax name and a transfer syntax list.

The transfer syntax list contains the names of those transfer syntaxes (or the names of specifications producing such transfer syntaxes) that the initiating PPM is capable of supporting for the named abstract syntax on the presentation-connection (at least one transfer syntax name for each proposed presentation context).

All presentation context identifiers contained in this parameter shall be different and shall be odd integers.

Note – The presentation context identifiers are specified here to be odd integers so that they are chosen from a separate number space from those identifiers allocated by the responding PPM (see also § 6.5).

See also § 6.2.6.1.

6.2.2.8 *Default context name*

This shall be the Default context name parameter from the P-CONNECT request service primitive and shall appear as the Default context name parameter of the P-CONNECT indication service primitive, if issued. It contains two components: an abstract syntax name and a transfer syntax name (or the name of a specification producing such a transfer syntax). The transfer syntax name component identifies the transfer syntax required by the initiating PPM for the default context to be used on the presentation-connection.

See also § 6.2.6.2.

6.2.2.9 *Quality of service*

This shall be the Quality of service parameter from the P-CONNECT request service primitive and shall appear as the Quality of service parameter of the P-CONNECT indication service primitive, if issued.

6.2.2.10 *Presentation requirements*

This shall be the Presentation requirements parameter from the P-CONNECT request service primitive and shall identify the presentation functional units proposed by the initiating PS-user in the P-CONNECT request service primitive. It shall appear as the Presentation requirements parameter of the P-CONNECT indication service primitive, if issued, unless the responding PPM does not support all of them, in which case only those functional units supported by the responding PPM shall appear.

See also § 6.2.6.3.

6.2.2.11 *User session requirements*

This shall be the Session requirements parameter from the P-CONNECT request service primitive and shall identify the requirements to the underlying session-service proposed by the PS-user. It shall appear as the Session requirements parameter of the P-CONNECT indication service primitive, if issued.

6.2.2.12 *Revised session requirements*

This shall be the Session requirements parameter from the P-CONNECT request service primitive, supplemented by such additional requirements as are needed to support the presentation protocol.

6.2.2.13 *Initial synchronization point serial number*

This shall be the Initial synchronization point serial number parameter from the P-CONNECT request service primitive, and shall appear as the Initial synchronization point serial number parameter of the P-CONNECT indication service primitive, if issued.

6.2.2.14 *Initial assignment of tokens*

This shall be the Initial assignment of tokens parameter from the P-CONNECT request service primitive, and shall appear as the Initial assignment of tokens parameter of the P-CONNECT indication service primitive, if issued.

6.2.2.15 *Session connection identifier*

This shall be the Session connection identifier parameter from the P-CONNECT request service primitive, and shall appear as the Session connection identifier parameter of the P-CONNECT indication service primitive, if issued.

6.2.2.16 *User data*

This shall represent the User data parameter from the P-CONNECT request service primitive, and shall appear as the User data parameter of the P-CONNECT indication service primitive, if issued. If the Presentation context definition list parameter is not present, then it shall be a list of presentation data values (including any embedded presentation data values) from the default context. Otherwise it shall be a list of presentation data values (including any embedded presentation data values) from presentation contexts proposed in the Presentation context definition list parameter.

6.2.3 *CPA PPDU associated parameters*

An instance of a CPA PPDU need not contain values for all possible parameters; in addition to the Responding-presentation-selector and Responding-session-address parameters, it should only contain values for equivalent parameter value present in the CP PPDU for which it is a reply.

6.2.3.1 *Mode selector*

This shall be the Mode selector parameter from the CP PPDU.

6.2.3.2 *Protocol version*

This shall identify the version of the presentation protocol selected for use on this presentation-connection. The version of the protocol defined in this Recommendation shall be version-1.

6.2.3.3 *Responding-presentation-selector*

This shall be the presentation-selection part of the Responding-presentation-address parameter from the P-CONNECT response service primitive and shall appear as the responding-presentation-selector part of the Responding-presentation-address parameter of the P-CONNECT confirm service primitive.

6.2.3.4 *Responding-session-address*

This shall be the session-address part of the Responding-presentation-address parameter from the P-CONNECT response service primitive and shall appear as the session-address part of the Responding-presentation-address parameter of the P-CONNECT confirm service primitive.

See also § 6.2.6.4.

6.2.3.5 *Presentation context definition result list*

This shall represent the Presentation context definition result list parameter of the P-CONNECT response service primitive and shall appear as the Presentation context definition result list parameter of the P-CONNECT confirm service primitive. It consists of a list containing the same number of items as the Presentation context definition list parameter of the CP PPDU. Each item shall be a reply to the corresponding item in the CP PPDU and contains one or two components, a presentation context definition result and an optional component which is either a transfer syntax name (or the name of a specification producing such a transfer syntax) or a provider reason.

The presentation context definition result shall take one of the values:

- "acceptance";
- "user-rejection";
- "provider rejection".

The transfer syntax name shall be present if the presentation context definition result takes the value "acceptance". It shall be one of the names proposed by the initiating PPM as possible transfer syntaxes for the identified presentation context, and shall determine the transfer syntax which the responding PPM has selected.

The provider reason shall be present if the presentation context definition result component takes the value "provider-rejection". It specifies the reason for rejection of the presentation context definition by the responding PPM and shall take one of the values:

- reason not specified;
- abstract syntax requirements not supported;
- proposed transfer syntaxes not supported;
- local limit on DCS exceeded.

See also § 6.2.6.1.

6.2.3.6 *Quality of service*

This shall be the Quality of service parameter from the P-CONNECT response service primitive and shall appear as the Quality of service parameter of the P-CONNECT confirm service primitive.

6.2.3.7 *Presentation requirements*

This shall be the Presentation requirements parameter from the P-CONNECT response service primitive. It shall appear as the Presentation requirements parameter of the P-CONNECT confirm service primitive.

See also § 6.2.3.6.

6.2.3.8 *User session requirements*

This shall be the Session requirements parameter from the P-CONNECT response service primitive and shall appear as the Session requirements parameter of the P-CONNECT confirm service primitive.

6.2.3.9 *Revised session requirements*

This shall be the Session requirements parameter from the P-CONNECT response service primitive, supplemented by such additional requirements as are needed to support the presentation protocol.

6.2.3.10 *Initial synchronization point serial number*

This shall be the initial synchronization point serial number parameter from the P-CONNECT response service primitive, and shall appear as the Initial synchronization point serial number parameter of the P-CONNECT confirm service primitive.

6.2.3.11 *Initial assignment of Tokens*

This shall be the Initial assignment of tokens parameter from the P-CONNECT response service primitive, and shall appear as the Initial assignment of tokens parameter of the P-CONNECT confirm service primitive.

6.2.3.12 *Session Connection Identifier*

This shall be the Session connection identifier parameter from the P-CONNECT response service primitive, and shall appear as the Session connection identifier parameter of the P-CONNECT confirm service primitive.

6.2.3.13 *User data*

This shall be the User data parameter from the P-CONNECT response service primitive, and shall appear as the User data parameter of the P-CONNECT confirm service primitive. The rules of § 6.1.2 shall apply.

6.2.4 *CPR PPDU associated parameters*

An instance of a CPR PPDU need not contain values for all possible parameters; in addition to presentation-selector parameters it should only contain values for corresponding parameter values present in the CP PPDU for which it is a reply.

6.2.4.1 *Protocol version*

This shall identify each version of the presentation protocol that the responding PPM supports. The version of the protocol defined in this Recommendation shall be version-1.

See also § 6.2.6.4.

6.2.4.2 *Responding-presentation-selector*

This shall be the presentation-selector part of the Responding-presentation-address parameter from the P-CONNECT response service primitive and shall appear as the responding-presentation-selector part of the Responding-presentation-address parameter of the P-CONNECT confirm service primitive.

6.2.4.3 *Responding-session-address*

This shall be the session-address part of the Responding-presentation-address parameter from the P-CONNECT response service primitive and shall appear as the session-address part of the Responding-presentation-address parameter of the P-CONNECT confirm service primitive.

6.2.4.4 *Presentation context definition result list*

This shall be the Presentation context definition result list parameter of the P-CONNECT response service primitive and shall appear as the Presentation context definition result list parameter of the P-CONNECT confirm service primitive. It consists of a list containing the same number of items as the Presentation context definition list parameter of the CP PPDU. Each item shall be a reply to the corresponding item in the CP PPDU and contains one or two components, a presentation context definition result and an optional component which is either a transfer syntax name (or the name of a specification producing such a transfer syntax) or a provider reason.

See also § 6.2.6.1.

The presentation context definition result shall take one of the values:

- "acceptance";
- "user-rejection";
- "provider-rejection".

The transfer syntax name shall be present if the presentation context definition result takes the value "acceptance". It shall be one of the names proposed by the initiating PPM as possible transfer syntaxes for the identified presentation context, and shall determine the transfer syntax which the responding PPM has selected.

The provider reason shall be present if the presentation context definition result takes the value "provider-rejection". It specifies the reason for rejection of the presentation context definition by the responding PPM and shall take one of the values:

- reason not specified;
- abstract syntaxt requirements not supported;
- proposed transfer syntaxes not supported;
- local limit on DCS exceeded.

6.2.4.5 *Default context result*

This shall be the Default context result parameter of the P-CONNECT response service primitive and shall appear as the Default context result parameter of the P-CONNECT confirm service primitive. It may take the value "acceptance", "provider-rejection" or "user-rejection".

See also § 6.2.6.2.

6.2.4.6 *Quality of service*

This shall be the Quality of service parameter from the P-CONNECT response service primitive, or if the CPR PPDU is initiated by the responding PPM on receipt of a CP PPDU it shall be supplied by the responding PPM. In either case it shall appear as the Quality of service parameter of the P-CONNECT confirm service primitive. It shall identify a quality of service required by the responding PS-user or presentation-service-provider.

6.2.4.7 *Session requirements*

This shall be the Session requirements parameter from the P-CONNECT response service primitive, or if the CPR PPDU is initiated by the responding PPM on receipt of a CP PPDU it shall be supplied by the responding PPM. In either case it shall appear as the Session requirements parameter of the P-CONNECT confirm service primitive. It shall identify those session functional units required by the responding PS-user or presentation-service-provider.

Note – If the presentation-connection establishment proposal is rejected by the PS-user, this parameter shall represent the session requirements of the PS-user as indicated in the response primitive; there is no Revised session requirements parameter in this PPDU.

6.2.4.8 *Session connection identifier*

This shall be the Session connection identifier parameter from the P-CONNECT response service primitive, and shall appear as the Session connection identifier parameter of the P-CONNECT confirm service primitive.

6.2.4.9 *Provider reason*

If present, this shall indicate that the rejection is by the responding presentation-service-provider; if absent, this shall indicate that the rejection is by the responding PS-user. This parameter shall indicate the reason for the rejection of the presentation-connection establishment proposal and shall appear as the Provider reason parameter of the P-CONNECT confirm service primitive. It shall take one of the following values:

- reason not specified (transient);
- temporary congestion (transient);
- local limit exceeded (permanent);
- called-presentation-address unknown (permanent);
- protocol version not supported (permanent);
- default context not supported (permanent);
- user data not readable (permanent);
- no PSAP available from the set of PSAPs identified by the called-presentation-address (transient).

6.2.4.10 *User data*

This shall be the User data parameter from the P-CONNECT response service primitive, and shall appear as the User data parameter of the P-CONNECT confirm service primitive. This parameter shall contain encodings of presentation data values (including any embedded presentation data values) from transfer syntaxes contained in the Presentation context definition result list parameter of this CPR PPDU, if present, or otherwise according to the default context. It is not present if the presentation-connection establishment proposal is rejected by the presentation-service-provider.

6.2.5 *Procedure*

6.2.5.1 When a P-CONNECT request service primitive is received by a PPM (the initiator), it shall initiate the establishment of a presentation-connection by sending a CP PPDU containing the presentation data values and proposed parameters necessary for the operation of the presentation-connection (see § 6.2.2).

6.2.5.2 As an initiator's option, the presentation data values contained in a CP PPDU may be encoded more than once to allow the transfer of the same presentation data values using a number of different transfer syntaxes.

6.2.5.3 The responding PPM is not required to examine more than one encoding for each presentation data value received. If, for any presentation data value received, all its examined encodings are expressed according to transfer syntaxes not supported by the responding PPM, then the responding PPM shall refuse the proposed presentation-connection by sending a CPR PPDU with a Provider reason parameter value of "user data not readable".

6.2.5.4 If the initiating PPM is unable to establish a presentation-connection due to an inability to establish a session-connection, it shall issue a P-CONNECT confirm service primitive with a Result parameter value of a "provider-rejection" and the presentation-connection shall not be established.

6.2.5.5 The responding PPM may refuse the proposed presentation-connection (if, for example, the parameter values of the CP PPDU are unacceptable; see also § 6.2.6), in which case it shall send a CPR PPDU with a Provider reason parameter included (see § 6.2.4). Alternatively, if not refusing, it shall issue a P-CONNECT indication service primitive.

6.2.5.6 If the responding PPM then receives a P-CONNECT response service primitive with a Result parameter value of "user-rejection", it shall send a CPR PPDU (see § 6.2.4), but if it receives a P-CONNECT response service primitive with a Result parameter value of "acceptance", it shall send a CPA PPDU (see § 6.2.3).

6.2.5.7 If the initiating PPM receives CPR PPDU refusing the presentation-connection, then it shall issue a P-CONNECT confirm service primitive with a Result parameter value of "user-rejection" (if the Provider reason parameter is not present) or "provider-rejection" (if the Provider reason parameter is present), and the presentation-connection shall not be established.

6.2.5.8 If the initiating PPM receives a CPA PPDU accepting the presentation-connection, then it shall issue a P-CONNECT confirm service primitive with a Result parameter value of "acceptance", and the presentation-connection shall be established.

6.2.5.9 If the presentation-connection is established, the DCS of each PPM is set according to the parameters of the CPA PPDU.

6.2.6 *Negotiation*

6.2.6.1 *Presentation context negotiation*

The DCS determined during presentation-connection establishment is negotiated between the peer PPMs and the PS-users.

The initiating PPM provides for each abstract syntax requested by its PS-user a list of transfer syntaxes it is capable of supporting for the presentation-connection. The responding PPM indicates in the P-CONNECT indication service primitive to its PS-user those abstract syntaxes it cannot support using one of the proposed transfer syntaxes, marking them as refused ("provider-rejection"). The responding PS-user indicates those abstract syntaxes it accepts or refuses in the P-CONNECT response service primitive. The responding PPM selects one item of the transfer syntax list as the transfer syntax to be used on the presentation-connection for each accepted presentation context.

A presentation context is identified by a presentation context identifier provided by the initiating PPM.

6.2.6.2 *Default context negotiation*

If the Default context name parameter is not present in the P-CONNECT request service primitive, then the interpretation of presentation data values from the default context is specified in a manner which is outside the scope of this Recommendation.

If the Default context name parameter is present and the responding PPM does not support the named default context, it shall send a CPR PPDU with a Provider reason parameter value of "default context not supported" and a Default context result parameter of value "provider-rejection".

If the responding PPM supports the named default context but receives a P-CONNECT response service primitive with a Default context result parameter value of "user-rejection", then it shall send a CPR PPDU with a Default context result parameter of "user-rejection".

6.2.6.3 *Functional units negotiation*

Presentation functional units are negotiated between the two PS-users. The presentation functional units selected for the presentation-connection are those which are required by both PS-users and are supported by both PPMs. The negotiation of session functional units is subject to the rules of the Session Service Definition (Recommendation X.215).

6.2.6.4 *Protocol version negotiation*

Presentation protocol version is negotiated between the two PPMs.

In the CP PPDU, the initiating PPM provides a list of versions that it is capable of supporting. In the CPA PPDU, the responding PPM indicates the version of the presentation protocol used on the presentation-connection; this shall be one of the versions proposed by the initiating PPM. In the CPR PPDU, the responding PPM may indicate a list of versions that it is capable of supporting; the use of this list is a local matter.

6.2.7 *Collisions and interactions*

6.2.7.1 *P-U-ABORT*

If the initiating PPM receives a P-U-ABORT request service primitive after it has sent a CP PPDU but before it has issued a P-CONNECT confirm service primitive, it shall send an ARU PPDU and the presentation-connection shall not be established.

6.7.7.2 *ARU PPDU, ARP PPDU and S-P-ABORT*

If the initiating PPM receives an S-P-ABORT indication service primitive or an ARP PPDU, it shall issue a P-P-ABORT indication service primitive and the presentation-connection shall not be established.

If the initiating PPM receives an ARU PPDU, it shall issue a P-U-ABORT indication service primitive and the presentation-connection shall not be established.

The responding PPM shall react to ARU PPDUs, ARP PPDUs and S-P-ABORT indication service primitives as above, once it has issued a P-CONNECT indication service primitive.

6.3 *Normal release of connection*

6.3.1 *Purpose*

The procedure for the normal release of a presentation-connection is used by a PPM to release the presentation-connection without loss of data in transit.

6.3.2 *Procedure*

6.3.2.1 Normal release of the presentation-connection takes place concurrently with the release of the underlying session-connection. PPDUs are not explicitly defined, but implicitly given by the description of mapping in Section 7.

6.3.2.2 The SS-user data parameters of session-service primitives used shall represent or be represented by the User data parameters of the associated presentation-service primitives and shall be from presentation contexts as specified in § 6.1.2.

6.4 *Abnormal release of connection*

6.4.1 *Purpose*

The procedure for the abnormal release of a presentation-connection is used at any time to force the release of the presentation-connection. It is invoked by the P-U-ABORT service or in response to a protocol error or the reception of an invalid PPDU.

The procedure uses the following PPDUs:

a) ARU PPDU;

b) ARP PPDU.

6.4.2 *ARU PPDU associated parameters*

6.4.2.1 *Presentation context identifier list*

This parameter shall be present if the User data parameter is present in the ARU PPDU and if the context management functional unit has been selected, or if the Presentation context definition list parameter was present in the CP PPDU. For each presentation context used in the ARU PPDU User data parameter, this parameter identifies the transfer syntax used.

It consists of a list, each item of which contains two components, a presentation context identifier and an associated transfer syntax name (or the name of a specification producing such a transfer syntax).

Note – If the DCS is empty, this parameter shall be empty.

6.4.2.2 *User-data*

This parameter shall represent the User data parameter from the P-U-ABORT request service primitive and shall be represented by the User data parameter of the P-U-ABORT indication service primitive. The parameter shall be from presentation contexts as defined in § 6.1.2.

Note – If the length restrictions imposed by the underlying session-service prevent the inclusion of the presentation data values of the User data parameter in the SS-user data parameter of the S-U-ABORT request session-service primitive, the User data parameter will not be included in the ARU PPDU sent. The way in which the PPM is made aware of this is a local matter.

6.4.3 *ARP PPDU associated parameters*

6.4.3.1 *Provider Reason*

This parameter shall indicate one of the following reasons:

a) reason not specified;

b) unrecognized PPDU;

c) unexpected PPDU;

d) unexpected session-service primitive;

e) unrecognized PPDU parameter;

f) unexpected PPDU parameter;

g) invalid PPDU parameter value.

In cases c), d), e), f) and g), the Event identifier parameter shall also be present.

6.4.3.2 *Event identifier*

This parameter shall identify the PPDU or the session-service primitive which triggered the abort procedure.

6.4.4 *Procedure*

The procedure shall depend on the stimulus as follows:

6.4.4.1 *P-U-ABORT*

When a PPM receives a P-U-ABORT request service primitive and either:

a) a presentation-connection has been established; or

b) a CP PPDU has been sent, and neither a CPA PPDU nor a CPR PPDU has been received,

it shall send an ARU PPDU and the presentation-connection shall be released.

6.4.4.2 *Protocol error*

When a PPM receives an unrecognized or unexpected PPDU, or an unexpected session-service primitive, it shall issue a P-P-ABORT indication service primitive and, if possible, send an ARP PPDU. The presentation-connection shall be released.

6.4.4.3 *Invalid PPDU*

When a PPM receives a PPDU, containing an invalid PPDU parameter value or an unrecognized or unexpected PPDU parameter, including a PPDU with an unexpected presentation context identifier, or one for which the received bitstring does not represent a valid presentation data value (including any embedded presentation data value) in the corresponding abstract syntax, it shall issue a P-P-ABORT indication service primitive and send an ARP PPDU, if possible. The presentation-connection shall be released.

6.4.4.4 *S-P-ABORT*

When a PPM receives an S-P-ABORT indication session-service primitive, it shall issue a P-P-ABORT indication service primitive and the presentation-connection shall be released.

6.4.4.5 *ARU PPDU*

When a PPM receives an ARU PPDU it shall issue a P-U-ABORT indication service primitive and the presentation-connection shall be released.

6.4.4.6 *ARP PPDU*

When a PPM receives an ARP PPDU, it shall issue a P-P-ABORT indication service primitive and the presentation-connection shall be released.

Note – When the abnormal release procedure is applied during an attempt to establish a presentation-connection, the presentation-connection shall not be established.

6.4.5 *Collisions and Interactions*

The abnormal release procedure may be used at any time when a presentation-connection has been established or during presentation-connection establishment.

6.5 *Context Alteration*

6.5.1 *Purpose*

The context alteration procedure is used to modify the DCS. It negotiates the definition of one or more new presentation contexts to be added to the DCS, and also the deletion of presentation contexts which are members of the DCS. It is used by a requesting entity which has received a P-ALTER-CONTEXT request service primitive.

The procedure uses the following PPDUs:

a) AC PPDU,

b) ACA PPDU.

6.5.2 *AC PPDU associated parameters*

6.5.2.1 *Presentation context addition list*

This consists of a list containing one or more items. Each item represents one item of the Presentation context addition list parameter from the P-ALTER-CONTEXT request service primitive and shall be represented by one item of the Presentation context addition list parameter of the P-ALTER-CONTEXT indication service primitive. Each item contains three components, a presentation context identifier, an abstract syntax name, and a transfer syntax list. The transfer syntax list contains those transfer syntax names (or the names of specifications producing such transfer syntaxes) the requesting PPM is capable of supporting for the named abstract syntax. All presentation context identifiers contained in this parameter shall be different from the presentation context identifiers of all presentation contexts in the DCS or previously used in any PPDU on the presentation-connection. If the sending PPM is the initiator, all presentation context identifiers shall be odd integers, otherwise all shall be even integers.

6.5.2.2 *Presentation context deletion list*

This shall be the Presentation context deletion list parameter from the P-ALTER-CONTEXT request service primitive and shall appear as the Presentation context deletion list parameter of the P-ALTER-CONTEXT indication service primitive.

6.5.2.3 *User data*

This parameter shall represent the User data parameter of the P-ALTER-CONTEXT request service primitive and shall be represented by the User data parameter of the P-ALTER-CONTEXT indiction service primitive. This parameter shall be from presentation contexts as specified in § 6.1.2.

6.5.3 *ACA PPDU associated parameters*

6.5.3.1 *Presentation context addition result list*

This shall represent the Presentation context addition result list parameter of the P-ALTER-CONTEXT response service primitive and shall be represented by the Presentation context addition result list parameter of the P-ALTER-CONTEXT confirm service primitive. It consists of a list containing the same number of items as the Presentation context addition list parameter of the AC PPDU. Each item shall be a reply to the corresponding item in the AC PPDU and contains one or two components, a presentation context addition result and an optional component which is either a transfer syntax name (or the name of a specification producing such a transfer syntax) or a provider reason.

The presentation context addition result shall take one of the values:

- "acceptance";
- "user-rejection";
- "provider-rejection".

The transfer syntax name shall be present if the presentation context addition result takes the value "acceptance". It shall be one of the names proposed by the requesting PPM as possible transfer syntaxes for the identified presentation context, and shall determine the transfer syntax which the accepting PPM has selected.

The provider reason shall be present if the presentation context addition result takes the value "provider-rejection". It specifies the reason for rejection of the presentation context addition by the accepting PPM and shall take one of the values:

- reason not specified;
- abstract syntax not supported;
- proposed transfer syntaxes not supported;
- local limit on DCS exceeded.

6.5.3.2 *Presentation context deletion result list*

This shall be the Presentation context deletion result list parameter of the P-ALTER-CONTEXT response service primitive and shall appear as the Presentation context deletion result list parameter of the P-ALTER-CONTEXT confirm service primitive. It consists of a list containing the same number of items as the Presentation context deletion list parameter of the AC PPDU. Each item shall refer to the corresponding item in the AC PPDU and shall take one of the values:

- "acceptance";
- "user-rejection".

6.5.3.3 *User data*

This parameter shall represent the User data parameter of the P-ALTER-CONTEXT response service primitive and shall be represented by the user data parameter of the P-ALTER-CONTEXT confirm service primitive. This parameter shall be from presentation contexts as specified in § 6.1.2.

6.5.4 *Procedure*

6.5.4.1 When a P-ALTER-CONTEXT request service primitive is received by a PPM (the requestor), it shall send an AC PPDU.

Note – Those presentation contexts proposed for deletion are still available for presentation data values in the User data parameter of the AC PPDU.

6.5.4.2 When an AC PPDU is received by a PPM (the acceptor). It may itself refuse some or all of the proposed presentation context additions. It shall issue a P-ALTER-CONTEXT indication service primitive in which it shall mark refused addition proposals with the value "provider-rejection".

6.5.4.3 When a P-ALTER-CONTEXT response service primitive is received by the accepting PPM, it shall send an ACA PPDU indicating the acceptance or rejection of each proposed presentation context addition and of each proposed presentation context deletion.

6.5.4.4 When a P-ALTER-CONTEXT response service primitive is received by the accepting PPM, the presentation contexts proposed for addition and marked with "acceptance" shall be added to the DCS and be available for use from the time of receipt of the response, and may also be used for presentation data values contained in the User data parameter of the ACA PPDU. The presentation contexts proposed for deletion and marked with "acceptance" shall be deleted from the DCS and no longer available for use from the time of receipt of the response, and shall not be used for presentation data values contained in the User data parameter of the ACA PPDU.

6.5.4.5 When an ACA PPDU is received by the requesting PPM, it shall issue a P-ALTER-CONTEXT confirm service primitive.

6.5.4.6 When an ACA PPDU is received by the requesting PPM, the presentation contexts accepted in the ACA PPDU shall be added to the DCS and be available for use from the time of receipt of the ACA PPDU, and shall be accepted for presentation data values contained in the User data parameter of the ACA PPDU itself. The presentation contexts accepted for deletion in the ACA PPDU shall be deleted from the DCS and no longer be available for use from the time of receipt of the ACA PPDU.

6.5.5 *Collisions and interactions*

6.5.5.1 *AC PPDU*

Simultaneous P-ALTER-CONTEXT request service primitives by both PS-users shall be treated independently by the PPMs. Independent treatment of the simultaneous deletion requests applies even if the two PS-users have specified the same presentation context for removal from the DCS.

As a result of the independent treatment of the two requests, and the freedom of each PS-user to either accept or reject a proposal for presentation context deletion by the peer PS-user, a PPM must be prepared for the following cases, which shall not be treated as errors:

a) Receipt of a P-ALTER-CONTEXT response service primitive specifying deletion of a presentation context which is not a member of the DCS, but responding to a P-ALTER-CONTEXT indication service primitive; in this case, the PPM shall send an ACA PPDU using the value of the Presentation context deletion result list parameter of the P-ALTER-CONTEXT response service primitive;

b) Receipt of an ACA PPDU specifying deletion of a presentation context which is not a member of the DCS, but responding to an AC PPDU; in this case, the PPM shall issue a P-ALTER-CONTEXT confirm service primitive with the corresponding Presentation context deletion result list parameter value.

6.5.5.2 *P-U-ABORT, ARU PPDU, ARP PPDU and S-P-ABORT*

See § 6.4.

6.5.5.3 *Destructive session services*

If the sender of an AC PPDU receives an RS PPDU, or an S-U-EXCEPTION-REPORT, S-P-EXCEPTION-REPORT, S-ACTIVITY-DISCARD or S-ACTIVITY-INTERRUPT indication service primitive before it has received an ACA PPDU, it shall not issue a P-ALTER-CONTEXT confirm service primitive, and the PPM shall continue with the procedure as specified for the disrupting service or RS PPDU.

6.6 *Information transfer*

6.6.1 *Purpose*

The information transfer procedure is used to convey presentation data values (including any embedded presentation data values) originating from P-DATA, P-TYPED-DATA, P-CAPABILITY-DATA and P-EXPEDITED-DATA request service primitives, and P-CAPABILITY-DATA response service primitives.

The procedure uses the following PPDUs:

a) TD PPDU;

b) TTD PPDU;

c) TE PPDU;

d) TC PPDU;

e) TCC PPDU.

6.6.2 *PPDU associated parameters*

Each of the PPDUs used by this procedure has a single parameter.

6.6.2.1 *User data*

This parameter shall represent the User data parameter from the corresponding request or response service primitive and shall appear as the User data parameter of the corresponding indication or confirm service primitive, as appropriate. For the TE PPDU, the parameter contains presentation data values from the default context. For the TD, TTD, TC and TCC PPDUs, the parameter contains presentation data values (including any embedded presentation data values) from presentation contexts specified in § 6.1.2.

6.6.3 *Procedure*

6.6.3.1 When a P-DATA request service primitive is received by a PPM, it shall send a TD PPDU to transmit according to the agreed transfer syntaxes the presentation data values (including any embedded presentation data values) expressed in the P-DATA request service primitive. When a PPM receives a TD PPDU, it shall issue a P-DATA indication service primitive containing these presentation data values (including any embedded presentation data values).

6.6.3.2 When a P-TYPED-DATA request service primitive is received by a PPM, it shall send a TTD PPDU to transmit according to the agreed transfer syntaxes the presentation data values (including any embedded presentation data values) expressed in the P-TYPED-DATA request service primitive. When a PPM receives a TTD PPDU, it shall issue a P-TYPED-DATA indication service primitive containing these presentation data values (including any embedded presentation data values).

The TTD PPDU shall only be available if the session typed data functional unit has been proposed and selected in the User session requirements parameters of both the CP and CPA PPDUs.

6.6.3.3 When a P-EXPEDITED-DATA request service primitive is received by a PPM, it shall send a TE PPDU to transmit, according to the transfer syntax of the default context, the presentation data values (including any embedded presentation data values) expressed in the P-EXPEDITED-DATA request service primitive. When a PPM receives a TE PPDU, it shall issue a P-EXPEDITED-DATA indication service primitive containing these presentation data values (including any embedded presentation data values).

6.6.3.4 When a P-CAPABILITY-DATA request service primitive is received by a PPM, it shall send a TC PPDU to transmit according to the agreed transfer syntaxes the presentation data values (including any embedded presentation data values) expressed in the P-CAPABILITY-DATA request service primitive. When a PPM receives a TC PPDU, it shall issue a P-CAPABILITY-DATA indication service primitive containing these presentation data values (including any embedded presentation data values). If the accepting PPM then receives a P-CAPABILITY-DATA response service primitive, it shall send a TCC PPDU to transmit according to the agreed transfer syntaxes the presentation data values (including any embedded presentation data values) expressed in the P-CAPABILITY-DATA response primitive. When a PPM receives a TCC PPDU, it shall issue a P-CAPABILITY-DATA confirm service primitive containing these presentation data values (including any embedded presentation data values).

6.6.4 *Collisions and interactions*

6.6.4.1 *P-U-ABORT, ARU PPDU, ARP PPDU and S-P-ABORT*

See § 6.4.

6.7 *Token handling*

6.7.1 *Purpose*

The token handling procedure is used to make available to PS-users the token handling facilities of the session-service. It is used by a PPM to support the P-TOKEN-GIVE, P-TOKEN-PLEASE and P-CONTROL-GIVE request and indication service primitives.

6.7.2 *Procedure*

6.7.2.1 PPDUs are not explicitly defined, but implicitly given by the description of mapping in § 7.

6.7.2.2 The User data parameters of session-service primitives used shall represent or be represented by the User data parameters of the associated presentation-service primitives and shall be from presentation contexts as specified in § 6.1.2.

6.8 *Synchronization and resynchronization*

6.8.1 *Purpose*

The synchronization and resynchronization procedure are used to make available to PS-users the synchronization and resynchronization facilities of the session-service. They are used by a PPM to support the P-SYNC-MINOR, P-SYNC-MAJOR and P-RESYNCHRONIZE service primitives. The resynchronization procedure has influence on the DCS when the context restoration functional unit has been selected.

The procedure uses the following PPDUs:

a) RS PPDU;

b) RSA PPDU.

6.8.2 *RS PPDU associated parameters*

6.8.2.1 *Resynchronize type*

This shall be the Resynchronize type parameter from the P-RESYNCHRONIZE request service primitive and shall appear as the Resynchronize type parameter of the P-RESYNCHRONIZE indication service primitive.

6.8.2.2 *Synchronization point serial number*

This shall be the Synchronization point serial number parameter from the P-RESYNCHRONIZE request service primitive and shall appear as the Synchronization point serial number parameter of the P-RESYNCHRONIZE indication service primitive.

6.8.2.3 *Tokens*

This shall be the Tokens parameter from the P-RESYNCHRONIZE request service primitive and shall appear as the Tokens parameter of the P-RESYNCHRONIZE indication service primitive.

6.8.2.4 *Presentation context identifier list*

This consists of a list, each entry of which has two components, a presentation context identifier and an associated transfer syntax name. The list shall specify the DCS which results from the RS PPDU.

6.8.2.5 *User data*

This parameter shall represent the User data parameter of the P-RESYNCHRONIZE request service primitive and shall be represented by the User data parameter of the P-RESYNCHRONIZE indication service primitive. This parameter shall be from presentation contexts as specified in § 6.1.2.

6.8.3 *RSA PPDU associated parameters*

6.8.3.1 *Synchronization point serial number*

This shall be the Synchronization point serial number parameter from the P-RESYNCHRONIZE response primitive and shall appear as the Synchronization point serial number parameter of the P-RESYNCHRONIZE confirm service primitive.

6.8.3.2 *Tokens*

This shall be the Tokens parameter from the P-RESYNCHRONIZE response service primitive and shall appear as the Tokens parameter of the P-RESYNCHRONIZE confirm service primitive.

6.8.3.3 *Presentation context identifier list*

This consists of a list, each entry of which has two components, a presentation context identifier and an associated transfer syntax name. The list shall specify the DCS which results from the RSA PPDU.

6.8.3.4 *User data*

This parameter shall represent the User data parameter of the P-RESYNCHRONIZE response service primitive and shall be represented by the User data parameter of the P-RESYNCHRONIZE confirm service primitive. This parameter shall be from presentation contexts as specified in § 6.1.2.

6.8.4 *Procedure*

6.8.4.1 The syncpoint identifier of a service primitive is a syncpoint identifier where the value of the synchronization point serial number is equal to that of the corresponding parameter of that service primitive.

The resync identifier of a service primitive is a syncpoint identifier where the value of the synchronization point serial number is equal to that of the corresponding parameter of that service primitive minus one.

6.8.4.2 If a PPM receives a P-SYNC-MINOR request service primitive or issues a P-SYNC-MINOR indication service primitive and the context restoration functional unit has been selected, then it shall associate the current DCS with the syncpoint identifier of the request or indication service primitive.

6.8.4.3 If a PPM receives a P-SYNC-MAJOR response service primitive or issues a P-SYNC-MAJOR confirm service primitive and the context restoration functional unit has been selected, then it shall associate the current DCS with the syncpoint identifier of the response or confirm service primitive.

The PPM shall eliminate any associations between syncpoint identifiers and DCSs which it had previously made.

6.8.4.4 If a PPM receives a P-RESYNCHRONIZE request service primitive and the context management functional unit is not selected then it shall send a RS PPDU.

6.8.4.5 If a PPM receives a P-RESYNCHRONIZE request service primitive and the context management functional unit is selected but the context restoration functional unit is not selected then it shall send a RS PPDU. The Presentation context identifier list parameter shall correspond to the DCS known to the PPM.

6.8.4.6 If a PPM receives a P-RESYNCHRONIZE request primitive and the context restoration functional unit is selected then it shall set the DCS as follows:

a) If the resynchronize type is “abandon”, then the DCS is unchanged;

b) if the resynchronize type is “restart” or “set”, then:

   i) if the resync identifier of the primitive is associated with a DCS, then the DCS is restored to that associated with the resync identifier;

   ii) if the resync identifier of the primitive is less than each syncpoint identifier associated with a DCS, then the DCS is restored to that of the presentation-connection establishment;

   iii) in all other cases, the DCS is unchanged.

The PPM shall then send a RS PPDU with the Presentation context identifier list parameter value corresponding to the DCS.

6.8.4.7 If a PPM receives a RS PPDU and the context management functional unit is not selected, then it shall issue a P-RESYNCHRONIZE indication service primitive.

6.8.4.8 If a PPM receives a RS PPDU and the context management functional unit is selected, but the context restoration functional unit is not selected, then it shall, if an ACA PPDU is awaited, replace the DCS by that specified in the Presentation context identifier list parameter of the RS PPDU. It shall then (regardless of whether an ACA is awaited) issue a P-RESYNCHRONIZE indication service primitive.

6.8.4.9 If a PPM receives a RS PPDU and the context restoration functional unit is selected, then it shall set the DCS as follows:

a) if the resynchronize type is "abandon", then:

   i) if an ACA PPDU is awaited, then the DCS is replaced by that specified in the Presentation context identifier list parameter of the RS PPDU;

   ii) if an ACA PPDU is not awaited, then the DCS is unchanged;

b) if the resynchronize type is "restart" or "set", then:

   i) if the resync identifier of the service primitive is associated with a DCS, then the DCS is restored to that associated with the resync identifier;

   ii) if the resync identifier of the service primitive is less than each syncpoint identifier associated with a DCS, then the DCS is restored to that of the presentation-connection establishment;

   iii) in all other cases, the DCS is set as per (a) above.

The PPM shall then issue a P-RESYNCHRONIZE indication service primitive.

6.8.4.10 If a PPM receives a P-RESYNCHRONIZE response service primitive and the context management functional unit is not selected, it shall send a RSA PPDU.

6.8.4.11 If a PPM receives a P-RESYNCHRONIZE response service primitive and the context management functional unit is selected then it shall send a RSA PPDU. The Presentation context identifier list parameter shall correspond to the DCS known to the PPM.

6.8.4.12 If a PPM receives a RSA PPDU and the context management functional unit is not selected, then it shall issue a P-RESYNCHRONIZE confirm service primitive.

6.8.4.13 If a PPM receives a RSA PPDU and the context management functional unit is selected but the context restoration functional unit is not selected, then it shall replace the DCS by that specified in the Presentation context identifier list parameter of the PPDU. It shall then issue a P-RESYNCHRONIZE confirm primitive.

6.8.4.14 If a PPM receives a RSA PPDU and the context restoration functional unit is selected, then it shall set the DCS as follows:

a) if the resynchronize type is "abandon", then the DCS is replaced by that specified in the Presentation context identifier list parameter of the PPDU;

b) if the resynchronize type is "restart" or "set", and either there is no syncpoint identifier associated with a DCS or the resync identifier is not associated with a DCS and is greater than the lowest syncpoint identifier associated with a DCS, the DCS is replaced by that specified in the Presentation context identifier list parameter of the PPDU;

c) in all other cases, the DCS is unchanged.

The PPM shall then issue a P-RESYNCHRONIZE confirm service primitive.

Note – When receiving a RSA PPDU, the resynchronize type relevant (for the procedures described above) is the resynchronize type of the associated RS PPDU.

6.8.5 *Collisions and Interactions*

6.8.5.1 *P-U-ABORT, ARU PPDU, ARP PPDU, and S-P-ABORT*

See § 6.4.

6.8.5.2 *P-ALTER-CONTEXT, AC-PPDU and ACA PPDU*

See § 6.5.

6.9 *Exception reporting*

6.9.1 *Purpose*

The exception reporting procedure is used to make available to PS-users the exception reporting facilities of the session-service. It is used by a PPM to support the P-U-EXCEPTION-REPORT request and indication service primitive and the P-P-EXCEPTION-REPORT indication service primitive.

6.9.2 *Procedure*

6.9.2.1 PPDUs are not explicitly defined, but implicitly given by the description of mapping in § 7.

6.9.2.2 The User data parameters of session-service primitives used shall represent or be represented by the User data parameters of the associated presentation-service primitives and shall be from presentation contexts as specified in § 6.1.2.

6.10 *Activity management*

6.10.1 *Purpose*

The activity management procedure is used to make available to PS-users the activity management facilities of the session-service. It is used by a PPM to support the P-ACTIVITY-START and P-ACTIVITY-RESUME request and indication service primitives and the P-ACTIVITY-END, P-ACTIVITY-INTERRUPT and P-ACTIVITY-DISCARD request, indication, response and confirm service primitives.

When the context restoration functional unit is selected, the activity management procedure has influence on the DCS as defined by § 6.10.2.

6.10.2 *Procedure*

6.10.2.1 PPDUs are not explicitly defined, but implicitly given by the description of mapping in § 7.

6.10.2.2 The User data parameters of session-service primitives used shall represent or be represented by the User data parameters of the associated presentation-service primitives and shall be from presentation contexts as specified in § 6.1.2.

6.10.2.3 If a PPM receives a P-ACTIVITY-INTERRUPT response service primitive or issues a P-ACTIVITY-INTERRUPT confirm service primitive when an activity is in progress and the context restoration functional unit has been selected, then it shall replace the DCS with the inter-activity DCS.

6.10.2.4 If a PPM receives a P-ACTIVITY-DISCARD response service primitive, or issues a P-ACTIVITY-DISCARD confirm service primitive when an activity is in progress, and the context restoration functional unit has been selected, then it shall replace the DCS with the inter-activity DCS. It shall also eliminate any associations between syncpoint identifiers and DCSs which it had previously made.

6.10.2.5 If a PPM receives a P-ACTIVITY-END response service primitive, or issues a P-ACTIVITY-END confirm service primitive, and the context restoration functional unit has been selected, then it shall replace the DCS with the inter-activity DCS. It shall also eliminate any associations between syncpoint identifiers and DCSs which it had previously made.

6.10.2.6 If a PPM receives a P-ACTIVITY-RESUME request service primitive or issues a P-ACTIVITY-RESUME indication service primitive, where the Old session connection identifier parameter is absent in the request or indication service primitive, then it shall take the following actions:

a) it shall eliminate any associations between DCSs and pairs composed of the Old activity identifier parameter value of the request or indication service primitive and any synchronization point serial number greater than the value of the Synchronization point serial number parameter.

b) if the pair composed of the Old activity identifier and Synchronization point serial number parameter values and has a DCS associated with it, then it shall restore the DCS by that one.

7 Mapping of PPDUS onto the session-service

7.1 *Connection establishment*

7.1.1 *CP PPDU*

The CP PPDU shall be conveyed from the initiating PPM to the responding PPM in the S-CONNECT request and indication session-service primitives.

7.1.1.1 *CP PPDU associated parameters*

Table 1/X.226 defines the mapping of the CP PPDU associated parameters onto S-CONNECT parameters.

TABLE 1/X.226

Mapping of CP PPDU associated parameters onto S-CONNECT parameters

CP PPDU associated parameter	S-CONNECT parameter	m/nm/s
Mode selector	SS-user data	m
Protocol version	SS-user data	nm
Calling-presentation-selector	SS-user data	nm
Calling-session-address	Calling SSAP address	s
Called-presentation-selector	SS-user data	nm
Called-session-address	Called SSAP address	s
Presentation context definition list	SS-user data	nm
Default context name	SS-user data	nm
Quality of service	Quality of service	s
Presentation requirements	SS-user data	nm
User session requirements	SS-user data	nm
Revised session requirements	Session requirements	s
Initial synchronization point serial number	Initial synchronization point serial number	s
Initial assignment of tokens	Initial assignment of tokens	s
Session connection identifier	Session connection identifier	s
User data	SS-user data	nm

m: mandatory.

nm: non-mandatory.

s: as defined in the Session Service Definition (Recommendation X.215).

7.1.2 *CPA PPDU*

The CPA PPDU shall be conveyed from the responding PPM to the initiating PPM in the S-CONNECT response and confirm session-serivce primitive.

7.1.2.1 *CPA PPDU associated parameters*

Table 2/X.226 defines the mapping of the CPA PPDU associated parameters onto S-CONNECT parameters.

TABLE 2/X.226

Mapping of CPA PPDU associated parameters onto S-CONNECT parameters

CPA PPDU associated parameter	S-CONNECT parameter	m/nm/s
Mode selector	SS-user data	m
Protocol version	SS-user data	nm
Responding-presentation-selector	SS-user data	nm
Responding-session-address	Responding SSAP address	s
Presentation context definition result list	SS-user data	nm
Quality of service	Quality of service	s
Presentation requirements	SS-user data	nm
User session requirements	SS-user data	nm
Revised session requirements	Session requirements	s
Initial synchronization point serial number	Initial synchronization point serial number	s
Initial assignment of tokens	Initial assignment of tokens	s
Session connection identifier	Session connection identifier	s
User data	SS-user data	nm

m: mandatory.

nm: non-mandatory.

s: as defined in the Session Service Definition (Recommendation X.215).

7.1.2.2 *S-CONNECT Result parameter*

This parameter shall have the value “accept”.

7.1.3 *CPR PPDU*

The CPR PPDU shall be conveyed from the responding PPM to the initiating PPM in the S-CONNECT response and confirm session-service primitive.

However, when the session-service-provider rejects the session-connection establishment proposal, there is no explicit S-CONNECT response session-service primitive and corresponding CPR PPDU.

7.1.3.1 *CPR PPDU associated parameters*

Table 3/X.226 defines the mapping of the CPR PPDU associated parameters onto S-CONNECT parameters.

TABLE 3/X.226

Mapping of CPR PPDU associated parameters onto S-CONNECT parameters

CPR PPDU associated parameter	S-CONNECT parameter	m/nm/s
Protocol version	SS-user data	nm
Responding-presentation-selector	SS-user data	nm
Responding-session-address	Responding SSAP address	s
Presentation context definition result list	SS-user data	nm
Default context result	SS-user data	nm
Quality of service	Quality of service	s
Session requirements	Session requirements	s
Session connection identifier	Session connection identifier	s
Provider reason	SS-user data	nm
User data	SS-user data	nm

m: mandatory.

nm: non-mandatory.

s: as defined in the Session Service Definition (Recommendation X.215).

7.1.3.2 *S-CONNECT Result parameter*

This parameter may take the values:

- "reject by SS-provider" (a whole class of values);
- "reject by called SS-user" with SS-user data.

The former case arises when rejection is initiated by the session-service-provider; the Provider reason parameter is absent even though rejection is initiated by the presentation-service-provider. The latter case arises when rejection is initiated by the responding PPM or PS-user; the Provider reason parameter is present only if rejection is initiated by the responding PPM. The User data parameter of the PPDU may only be present when rejection is initiated by the responding PS-user.

7.2 *Normal release of connection*

Normal release of the presentation-connection takes place concurrently with normal release of the session-connection. Presentation-service primitives are mapped onto the corresponding session-service primitives. Table 4/X.226 defines the mapping.

TABLE 4/X.226

Mapping of normal release service primitives

Presentation primitive	Session primitive
P-RELEASE request	S-RELEASE request
P-RELEASE indication	S-RELEASE indication
P-RELEASE response	S-RELEASE response
P-RELEASE confirm	S-RELEASE confirm

7.3 *Abnormal release of connection*

7.3.1 *ARU PPDU*

The ARU PPDU shall be conveyed from the requesting PPM to the accepting PPM in the S-U-ABORT request and indication session-service primitives.

7.3.1.1 *ARU PPDU associated parameters*

Table 5/X.226 defines the mapping of the ARU PPDU associated parameters onto S-U-ABORT parameters.

TABLE 5/X.226

Mapping of ARU PPDU associated parameters onto S-U-ABORT parameters

ARU PPDU associated parameter	S-U-ABORT parameter	m/nm/s
Presentation context identifier list	SS-user data	nm
User data	SS-user data	nm

m: mandatory.

nm: non-mandatory.

s: as defined in the Session Service Definition (Recommendation X.215).

7.3.2 *ARP PPDU*

The ARP PPDU shall be conveyed from the requesting PPM to the accepting PPM in the S-U-ABORT request and indication session-service primitives.

7.3.2.1 *ARP PPDU associated parameters*

Table 6/X.226 defines the mapping of the ARP PPDU associated parameters onto S-U-ABORT parameters.

TABLE 6/X.226

Mapping of ARP PPDU associated parameters onto S-U-ABORT parameters

ARP PPDU associated parameter	S-U-ABORT parameter	m/nm/s
Provider reason	SS-user data	nm
Event identifier	SS-user data	nm

m: mandatory.

nm: non-mandatory.

s: as defined in the Session Service Definition (Recommendation X.215).

7.4 *Context Alteration*

7.4.1 *AC PPDU*

The AC PPDU shall be conveyed from the requesting PPM to the accepting PPM in the S-TYPED- DATA request and indication session-service primitives.

7.4.1.1 *AC PPDU associated parameters*

Table 7/X.226 defines the mapping of the AC PPDU associated parameters onto S-TYPED-DATA parameters.

7.4.2 *ACA PPDU*

The ACA PPDU shall be conveyed from the accepting PPM to the requesting PPM in the S-TYPED-DATA response and confirm session-service primitives.

7.4.2.1 *ACA PPDU associated parameters*

Table 8/X.226 defines the mapping of the ACA PPDU associated parameters onto S-TYPED-DATA parameters.

TABLE 7/X.226

Mapping of AC PPDU associated parameters onto S-TYPED-DATA parameters

AC PPDU associated parameter	S-TYPED-DATA parameter	m/nm/s
Presentation context addition list	SS-user data	nm
Presentation context deletion list	SS-user data	nm
User data	SS-user data	nm

m: mandatory.

nm: non-mandatory.

s: as defined in the Session Service Definition (Recommendation X.215).

TABLE 8/X.226

Mapping of ACA PPDU associated parameters onto S-TYPED-DATA parameters

ACA PPDU associated parameter	S-TYPED-DATA parameter	m/nm/s
Presentation context addition result list	SS-user data	nm
Presentation context deletion result list	SS-user data	nm
User data	SS-user data	nm

m: mandatory.

nm: non-mandatory.

s: as defined in the Session Service Definition (Recommendation X.215).

7.5 *Information Transfer*

7.5.1 *TTD PPDU*

The TTD PPDU shall be conveyed from the requesting PPM to the accepting PPM in the S-TYPED-DATA request and indication session-service primitives.

7.5.1.1 *TTD PPDU associated parameters*

Table 9/X.226 defines the mapping of the TTD PPDU associated parameters onto S-TYPED-DATA parameters.

TABLE 9/X.226

Mapping of TTD PPDU associated parameters onto S-TYPED-DATA parameters

TTD PPDU associated parameter	S-TYPED-DATA parameter	m/nm/s
User data	SS-user data	nm

m: mandatory.

nm: non-mandatory.

s: as defined in the Session Service Definition (Recommendation X.215).

7.5.2 *TD PPDU*

The User data parameter of a TD PPDU shall form the SS-user data parameter of an S-DATA request service primitive and corresponding indication service primitive.

7.5.3 *TE PPDU*

The User data parameter of a TE PPDU shall form the SS-user data parameter of an S-EXPEDITED-DATA request service primitive and corresponding indication service primitive.

7.5.4 *TC PPDU*

The User data parameter of the TC PPDU shall form the SS-user data parameter of an S-CAPABILITY-DATA request service primitive and corresponding indication service primitive.

7.5.5 *TCC PDU*

The User data parameter of the TCC PPDU shall form the SS-user data parameter of an S-CAPABILITY-DATA response service primitive and corresponding confirm service primitive.

7.6 *Token Handling*

Token handling services are provided by the underlying session-service. Presentation-service primitives are mapped onto the corresponding session-service primitives. Table 10/X.226 defines the mapping.

7.7 *Synchronization*

Synchronization services are provided by the underlying session-service. The presentation-service primitives are mapped onto the corresponding session-service primitives. Table 11/X.226 defines the mapping.

TABLE 10/X.226

Mapping of token handling service primitives

Presentation primitive	Session primitive
P-TOKEN-GIVE request	S-TOKEN-GIVE request
P-TOKEN-GIVE indication	S-TOKEN-GIVE indication
P-TOKEN-PLEASE request	S-TOKEN-PLEASE request
P-TOKEN-PLEASE indication	P-TOKEN-PLEASE indication
P-CONTROL-GIVE request	S-CONTROL-GIVE request
P-CONTROL-GIVE indication	S-CONTROL-GIVE indication

TABLE 11/X.226

Mapping of synchronization service primitives

Presentation primitive	Session primitive
P-SYNC-MINOR request	S-SYNC-MINOR request
P-SYNC-MINOR indication	S-SYNC-MINOR indication
P-SYNC-MINOR response	S-SYNC-MINOR response
P-SYNC-MINOR confirm	S-SYNC-MINOR confirm
P-SYNC-MAJOR request	S-SYNC-MAJOR request
P-SYNC-MAJOR indication	S-SYNC-MAJOR indication
P-SYNC-MAJOR response	S-SYNC-MAJOR response
P-SYNC-MAJOR confirm	S-SYNC-MAJOR confirm

7.8 *Resynchronization*

7.8.1 *RS PPDU*

The RSA PPDU shall be conveyed from the requesting PPM to the accepting PM in the S-RESYNCHRONIZE request and indication session-service primitives.

7.8.1.1 *RS PPDU associated parameters*

Table 12/X.226 defines the mapping of the RS PPDU associated parameters onto S-RESYNCHRONIZE parameters.

TABLE 12/X.226

Mapping of RS PPDU associated parameters onto S-RESYNCHRONIZE parameters

RS PPDU associated parameter	S-RESYNCHRONIZE parameter	m/nm/s
Resynchronize type	Resynchronize type	s
Synchronization point serial number	Synchronization point serial number	s
Tokens	Tokens	s
Presentation context identifier list	SS-user data	nm
User data	SS-user data	nm

m: mandatory.

nm: non-mandatory.

s: as defined in the Session Service Definition (Recommendation X.215).

7.8.2 *RSA PPDU*

The RSA PPDU shall be conveyed from the accepting PPM to the requesting PPM in the S-RESYNCHRONIZE response and confirm session-service primitives.

7.8.2.1 *RSA PPDU associated parameters*

Table 13/X.226 defines the mapping of the RSA PPDU associated parameters onto S-RESYNCHRONIZE parameters.

7.9 *Exception Reporting*

Exception reporting services are provided by the underlying session-service. Presentation-service primitives are mapped onto the corresponding session-service primitives. Table 14/X.226 defines the mapping.

TABLE 13/X.226

Mapping of RSA PPDU associated parameters onto S-RESYNCHRONIZE parameters

RSA PPDU associated parameter	S-RESYNCHRONIZE parameter	m/nm/s
Synchronization point serial number	Synchronization point serial number	s
Tokens	Tokens	s
Presentation context identifier list	SS-user data	nm
User data	SS-user data	nm

m: mandatory.

nm: non-mandatory.

s: as defined in the Session Service Definition (Recommendation X.215).

TABLE 14/X.226

Mapping of Exception reporting service primitives

Presentation primitive	Session primitive
P-P-EXCEPTION-REPORT indication	S-P-EXCEPTION-REPORT indication
P-U-EXCEPTION-REPORT request	S-U-EXCEPTION-REPORT request
P-U-EXCEPTION-REPORT indication	S-U-EXCEPTION-REPORT indication

7.10 *Activity Management*

Activity management services are provided by the underlying session-service. Presentation-service primitives are mapped onto the corresponding session-service primitives. Table 15/X.226 defines the mapping.

TABLE 15/X.226

Mapping of Activity management service primitives primitives

Presentation primitive	Session primitive
P-ACTIVITY-START request	S-ACTIVITY-START request
P-ACTIVITY-START indication	S-ACTIVITY-START indication
P-ACTIVITY-RESUME request	S-ACTIVITY-RESUME request
P-ACTIVITY-RESUME indication	S-ACTIVITY-RESUME indication
P-ACTIVITY-INTERRUPT request	S-ACTIVITY-INTERRUPT request
P-ACTIVITY-INTERRUPT indication	S-ACTIVITY-INTERRUPT indication
P-ACTIVITY-INTERRUPT response	S-ACTIVITY-INTERRUPT response
P-ACTIVITY-INTERRUPT confirm	S-ACTIVITY-INTERRUPT confirm
P-ACTIVITY-DISCARD request	S-ACTIVITY-DISCARD request
P-ACTIVITY-DISCARD indication	S-ACTIVITY-DISCARD indication
P-ACTIVITY-DISCARD response	S-ACTIVITY-DISCARD response
P-ACTIVITY-DISCARD confirm	S-ACTIVITY-DISCARD confirm
P-ACTIVITY-END request	S-ACTIVITY-END request
P-ACTIVITY-END indication	S-ACTIVITY-END indication
P-ACTIVITY-END response	S-ACTIVITY-END response
P-ACTIVITY-END confirm	S-ACTIVITY-END confirm

8 Structure and encoding of PPDUs

8.1 *General*

8.1.1 The struture of PPDUs (whether explicity defined or implicitly given) shall be defined by:

a) the mapping onto parameters of session-service primitives;

b) the structure of session-service primitive SS-user data parameter values.

8.1.2 The structure of SS-user data parameter values is specified using:

a) the notation ASN.1 (Recommendation X.208);

b) additional comments contained in the ASN.1 description;

Note – ASN.1 comments in § 8.2 are an integral part of this Recommendation, and frequently express requirements.

c) rules of extensibility as specified in § 8.5 when operating in normal mode.

8.1.3 The encoding of SS-user data parameter values is specified in § 8.3.

8.2 *Structure of SS-user data parameter values*

```
X.226-PRESENTATION DEFINITIONS :: =

BEGIN

--

-- In X.410-1984 mode, the value of the SS-user data parameter of
-- the S-CONNECT request and indication session-service primitives
-- shall be a CP-type value.

--

-- In normal mode, the value of the SS-user data parameter of the
-- S-CONNECT request and indication session-service primitives shall
-- be a CP-type value, followed as a requestor's option by zero or
-- more CPC-type values.


CP-type :: = SET
{       [0]   IMPLICIT Mode-selector
        [1]   IMPLICIT SET
              { COMPONENTS OF Reliable-Transfer-APDUs.RTORQapdu }                    OPTIONAL

              -- Shall be used for X.410 mode only. Shall be bitwise
              -- compatible with CCITT Recommendation X.410-1984.
              -- This shall be the User data parameter of
              -- the CP PPDU [1] --,


        [2]   IMPLICIT SEQUENCE
              {     [0] IMPLICIT    Protocol-version                         DEFAULT {version-1},
                    [1] IMPLICIT    Calling-presentation-selector                      OPTIONAL,
                    [2] IMPLICIT    Called-presentation-selector                       OPTIONAL,
                    [4] IMPLICIT    Presentation-context-definition-list               OPTIONAL,
                    [6] IMPLICIT    Default-context-name                               OPTIONAL,
                    [8] IMPLICIT    Presentation-requirements                          OPTIONAL,
                    [9] IMPLICIT    User-session-requirements                          OPTIONAL,

                    -- shall not be present if equal to the Revised session
                    -- requirements parameter --,

                                    User-data                                          OPTIONAL
              }                                                                    OPTIONAL

                    -- Shall be used for normal mode only.
                    -- Shall be the parameters of the CP PPDU.
                    -- As an initiator's option, the presentation data values
                    -- contained in a CP PPDU may be encoded more than once,
                    -- using CPC-type values, to allow the transfer of the same
                    -- presentation data values using a number of different
                    -- transfer syntaxes.

}
--
```

[1] ASN.1 module Reliable-Transfer-APDUs is defined in Recommendation X.228.

```
CPC-type ::= User-data

-- Shall be used for normal mode only.
-- Shall not be present if the Presentation context definition list
-- parameter is not present in the CP PPDU.
-- Each instance of this data type shall contain all of the presentation
-- data values which were contained in the User data parameter of
-- the CP PPDU
-- This shall be the same set of presentation data values which were
-- contained in the CP-type.
-- The SS-user data parameter value of the S-CONNECT
-- response and confirm session-service primitives shall be
-- a CPA-PPDU value when the Result parameter value
-- is "accept".

--


CPA-PPDU ::= SET

{       [0]   IMPLICIT Mode-selector
        [1]   IMPLICIT SET
              { COMPONENTS OF Reliable-Transfer-APDUs.RTOACapdu }                   OPTIONAL

              -- Shall be used for X.410 mode only. Shall be bitwise
              -- compatible with CCITT Recommendation X.410-1984.
              -- This shall be the User data parameter of
              -- the CPA PPDU 2) --,

        [2]   IMPLICIT SEQUENCE

        {     [0] IMPLICIT      Protocol-version                              DEFAULT {version-1},
              [3] IMPLICIT      Responding-presentation-selector                      OPTIONAL,
              [5] IMPLICIT      Presentation-context-definition-result-list          OPTIONAL,
              [8] IMPLICIT      Presentation-requirements                             OPTIONAL,
              [9] IMPLICIT      User-session-requirements                             OPTIONAL,

              -- shall not be present if equal to the Revised session
              -- requirements parameter --

                                User-data                                             OPTIONAL
        }                                                                          OPTIONAL

              -- Shall be used for normal mode only

}

--
--
-- The SS-user data parameter value of
-- the S-CONNECT response and confirm
-- session-service primitives shall be
-- a CPR-PPDU value when the Result parameter
-- value is "reject by SS-provider"
-- or "reject by called SS-user".

--
```

[2] ASN.1 module Reliable-Transfer-APDUs is defined in Recommendation X.228.

```
CPR-PPDU ::= CHOICE

{       . SET {   COMPONENTS OF Reliable-Transfer-APDUs.RTORJapdu }

                -- Shall be used for X.410 mode only. Shall be bitwise
                -- compatible with CCITT Recommendation X.410-1984.
                -- This shall be the User data parameter of
                -- the CPR PPDU 3) --,

                SEQUENCE {

                    [0] IMPLICIT    Protocol-version                            DEFAULT {version-1},
                    [3] IMPLICIT    Responding-presentation-selector                     OPTIONAL,
                    [5] IMPLICIT    Presentation-context-definition-result-list          OPTIONAL,
                    [7] IMPLICIT    Default-context-result                               OPTIONAL,
                    [10] IMPLICIT   Provider-reason                                      OPTIONAL,

                                    User-data                                            OPTIONAL

               }

                -- Shall be used for normal mode only. --

}

--
--

-- The S-user data parameter of the S-U-ABORT
-- request and indication service primitives shall be
-- an Abort-type value.

--


Abort-type ::= CHOICE  {   ARU-PPDU  -- for a P-U-ABORT--,

                           ARP-PPDU  -- for a P-P-ABORT --

                       }

--


ARU-PPDU ::= CHOICE

        { SET {   COMPONENTS OF Reliable-Transfer-APDUs.RTABapdu }

                -- Shall be used for X.410 mode only. Shall be bitwise
                -- compatible with CCITT Recommendation X.410-1984.
                -- This shall be the User data parameter of
                -- the ARU PPDU. 3) --,

        [0]     IMPLICIT SEQUENCE

                {   [0] IMPLICIT    Presentation-context-identifier-list                 OPTIONAL,
                                    User-data                                            OPTIONAL

                }
                    -- Shall be used for normal mode only.

}

--
```

[3] ASN.1 module Reliable-Transfer-APDUs is defined in Recommendation X.228.

```
ARP-PPDU :: = SEQUENCE

        { provider-reason
              [0] IMPLICIT          Abort-reason                                    OPTIONAL,
              [1] IMPLICIT          Event-identifier                                OPTIONAL

        }

--
--

-- The SS-user data parameter value of
-- the S-TYPED-DATA request and
-- indication service primitives shall be
-- a Typed-data-type value

--


Typed-data-type :: = CHOICE

{       acPPDU
              [0] IMPLICIT AC-PPDU

              -- P-ALTER-CONTEXT
              -- request and indication --

        acaPPDU
              [1] IMPLICIT ACA-PPDU

              -- P-ALTER-CONTEXT
              -- response and confirm --

        ttdPPDU
                                    User-data

              -- P-TYPED-DATA
              -- request and indication --

}

--


AC-PPDU :: = SEQUENCE

        {     [0] IMPLICIT          Presentation-context-addition-list              OPTIONAL,
              [1] IMPLICIT          Presentation-context-deletion-list              OPTIONAL,
                                    User-data                                       OPTIONAL

        }

--


ACA-PPDU :: = SEQUENCE

        {     [0] IMPLICIT          Presentation-context-addition-result-list       OPTIONAL,
              [1] IMPLICIT          Presentation-context-deletion-result-list       OPTIONAL,
                                    User-data                                       OPTIONAL

        }

--

-- The SS-user data parameter value of
-- the S-RESYNCHRONIZE request and
-- indication service primitives shall be
-- an RS-PPDU value.

--
```

```
RS-PPDU :: = SEQUENCE
        {     [0] IMPLICIT        Presentation-context-identifier-list                OPTIONAL,
                                  User-data                                           OPTIONAL
        }
--
-- The SS-user data parameter value of
-- the S-RESYNCHRONIZE response and
-- confirm service primitives shall be
-- an RSA-PPDU value.
--

RSA-PPDU :: = SEQUENCE
        {     [0] IMPLICIT        Presentation-context-identifier-list                OPTIONAL,
                                  User-data                                           OPTIONAL
        }
--
--
-- The SS-user data parameter values of
-- the S-DATA, S-CAPABILITY-DATA,
-- S-EXPEDITED-DATA request and
-- indication session-service primitives and
-- S-CAPABILITY-DATA
-- response and confirm session-service
-- primitives shall be of type User-data.
--
-- The SS-user data parameter values of
-- all other session-service
-- primitives not described above shall
-- be of type User-data.
--
--
--

Abort-reason :: = INTEGER {
            reason-not-specified                                                      (0),
            unrecognized-ppdu                                                         (1),
            unexpected-ppdu                                                           (2),
            unexpected-session-service-primitive                                      (3),
            unrecognized-ppdu-parameter                                               (4),
            unexpected-ppdu-parameter                                                 (5),
            invalid-ppdu-parameter-value                                              (6) }

Abstract-syntax-name :: = OBJECT IDENTIFIER

Called-presentation-selector :: = Presentation Selector

Calling-presentation-selector :: = Presentation Selector

Context-list :: = SEQUENCE OF SEQUENCE {
            Presentation-context-identifier,
            Abstract-syntax-name,
            SEQUENCE OF Transfer-syntax-name }
```

```
Default-context-name :: = SEQUENCE
            [0] IMPLICIT      Abstract-syntax-name,
            [1] IMPLICIT      Transfer-syntax-name }

Default-context-result :: = Result

Event-identifier :: = INTEGER {
            cp-PPDU                                                       ( 0),
            cpa-PPDU                                                      ( 1),
            cpr-PPDU                                                      ( 2),
            aru-PPDU                                                      ( 3),
            arp-PPDU                                                      ( 4),
            ac-PPDU                                                       ( 5),
            aca-PPDU                                                      ( 6),
            td-PPDU                                                       ( 7),
            ttd-PPDU                                                      ( 8),
            te-PPDU                                                       ( 9),
            tc-PPDU                                                       (10),
            tcc-PPDU                                                      (11),
            rs-PPDU                                                       (12),
            rsa-PPDU                                                      (13),
            s-release-indication                                          (14),
            s-release-confirm                                             (15),
            s-token-give-indication                                       (16),
            s-token-please-indication                                     (17),
            s-control-give-indication                                     (18),
            s-sync-minor-indication                                       (19),
            s-sync-minor-confirm                                          (20),
            s-sync-major-indication                                       (21),
            s-sync-major-confirm                                          (22),
            s-p-exception-report-indication                               (23),
            s-u-exception-report-indication                               (24),
            s-activity-start-indication                                   (25),
            s-activity-resume-indication                                  (26),
            s-activity-interrupt-indication                               (27),
            s-activity-interrupt-confirm                                  (28),
            s-activity-discard-indication                                 (29),
            s-activity-discard-confirm                                    (30),
            s-activity-end-indication                                     (31),
            s-activity-end-confirm                                        (32) }

Mode-Selector :: = SET {
            [0] IMPLICIT INTEGER {x410-1984-mode (0), normal-mode (1) } }

Presentation-context-addition-list :: = Context-list

Presentation-context-addition-result-list :: = Result-list

Presentation-context-definition-list :: = Context-list

Presentation-context-definition-result-list :: = Result-list

Presentation-context-deletion-list :: = SEQUENCE OF
            Presentation-context-identifier

Presentation-context-deletion-result-list :: = SEQUENCE OF INTEGER {
            Acceptance (0), user-rejection (1) }
```

```
Presentation-context-identifier :: = INTEGER

Presentation-context-identifier-list :: = SEQUENCE OF SEQUENCE {
          Presentation-context-identifier,
          Transfer-syntax-name }

Presentation-requirements :: = BIT STRING {
          Context-management (0), restoration (1) }

Presentation-selector :: = OCTET STRING

Protocol-version :: = BIT STRING {version-1 (0) }

Provider-reason :: = INTEGER {
          reason-not-specified                                                (0),
          temporary-congestion                                                (1),
          local-limit-exceeded                                                (2),
          called-presentation-address-unknown                                 (3),
          protocol-version-not-surprised                                      (4),
          default-context-not-supported                                       (5),
          user-data-not-readable                                              (6),
          no-PSAP-available                                                   (7) }

Responding-presentation-selector :: = Presentation-selector

Result :: = INTEGER {
          acceptance                                                          (0),
          user-rejection                                                      (1),
          provider-rejection                                                  (2) }

Result-list :: = SEQUENCE OF SEQUENCE {
          [0] IMPLICIT      Result,
          [1] IMPLICIT      Transfer-syntax-name                         OPTIONAL,
          [2] IMPLICIT INTEGER

provider-reason
          { reason-not-specified                                              (0),
          abstract-syntax-not-supported                                       (1),
          proposed-transfer-syntaxes-not-supported                            (2),
          local-limit-on-DCS-exceeded                                 (3)    OPTIONAL,
          }
}

Transfer-syntax-name :: = OBJECT IDENTIFIER

User-data :: = CHOICE {
          [APPLICATION 0]   IMPLICIT    Simply-encoded-data
          [APPLICATION 1]   IMPLICIT    Fully-encoded-data }
          -- § 8.4 defines when each of the two alternatives shall be used.

Simply-encoded-data :: = OCTET STRING
          -- See § 8.4.1.
```

```
Fully-encoded-data :: = SEQUENCE OF PVD-list

          -- Contains one or more PDV-list values.
          -- See § 8.4.2.


PDV-list :: = SEQUENCE {

          Transfer-syntax-name                                   OPTIONAL,
          Presentation-context-identifier,


presentation-data-values :: = CHOICE {

          single-ASN.1-type  [0]  ANY,
          octet-aligned      [1]  IMPLICIT OCTET STRING,
          arbitrary          [2]  IMPLICIT BIT STRING
          }

-- Contains one or more presentation data values from the same
-- presentation context.
-- See § 8.4.2.

}


User-session-requirements :: = BIT STRING {

          half-duplex                                            ( 0),
          duplex                                                 ( 1),
          expedited-data                                         ( 2),
          minor-synchronize                                      ( 3),
          major-synchronize                                      ( 4),
          resynchronize                                          ( 5),
          activity-management                                    ( 6),
          negotiated-release                                     ( 7),
          capability-data                                        ( 8),
          exceptions                                             ( 9),
          typed-data                                             (10) }

END
```

8.3 *Encoding of SS-user data parameter values*

8.3.1 Except for type User-data, ASN.1 datatypes specified in § 8.2 shall be encoded according to the Basic Encoding Rules for ASN.1 (Recommendation X.209).

8.3.2 The encoding of values of type User-data is specified in § 8.4.

8.3.3 The encoding of the SS-user data parameter of the S-CONNECT request and indication service primitives shall be the concatenation of the encodings of the CP-type value and the CPC-type values, if any.

8.4 *Encoding of values of type User-data*

8.4.1 *Simple encoding*

8.4.1.1 This encoding shall be used when the User-data value is of type Simply-encoded-data.

8.4.1.2 The User-data value shall be of type Simply-encoded-data when the default context is used.

8.4.1.3 The User-data value shall be of type Simply-encoded-data when the DCS contains only one member and the context management functional unit is not selected.

Note – This implies that simple encoding cannot be used in the User data parameter of the CP PPDU, except as in § 8.4.1.2.

8.4.1.4 Simple encoding shall be as follows:

a) The contents of the Simply-encoded-data value shall be the concatenation of the bitstrings[4] resulting from the encoding of the presentation data values forming the PS-user data value according to the appropriate transfer syntax.

b) Whenever User-data appears as an element of some other ASN.1 type in § 8.2, the encoding of the User-data value shall be according to the Basic Encoding Rules for ASN.1 (Recommendation X.209).

c) If (b) does not apply, the encoding of the User-data value shall be the contents octets of the Simply-encoded-data value (i.e. no identfier octets and no length octets) as specified in (a) above.

Note – When using simple encoding, the transfer syntax used shall either produce octet-aligned encodings or self-delimiting bitstrings (this is not the general case with transfer syntaxes).

8.4.2 *Full encoding*

8.4.2.1 This encoding shall be used when the User-data value is of type Fully-encoded-data.

8.4.2.2 The User-data value shall be of type Fully-encoded-data when the default context is not in use and:

a) the DCS contains more than one member; or

b) the context management functional unit has been selected.

8.4.2.3 The User-data value shall be of type Fully-encoded-data in the CP-type and CPC-type, except when the default context is in use.

8.4.2.4 Full encoding shall be the application of the Basic Encoding Rules for ASN.1 (Recommendation X.209) to the Fully-encoded-data value. The structure and contents of the presentation-data-values component of a PDV-list value shall be as specified in § 8.4.2.5.

8.4.2.5 The presentation-data-values component of a PDV-list value shall be encoded according to the Basic Encoding Rules for ASN.1 (Recommendation X.209). The various options for the presentation-data-values component of the PDV-list value shall be used as follows:

a) If the PDV-list value contains exactly one presentation data value which is a single ASN.1 type encoded according to the Basic Encoding Rules for ASN.1 (Recommendation X.209), then the option "single-ASN1-type" shall be used.

b) If the encodings of the presentation data values contained in the PDV-list value are each an integral number of octets and (a) does not apply, then the option "octet-aligned" shall be used. In this case, the contents octets of the OCTET STRING shall be the concatenation of the bitstrings resulting from the encoding of the presentation data values contained in that PDV-list value according to the appropriate transfer syntax.

c) If neither (a) nor (b) applies, the option "arbitrary" shall be used. The contents octets of the BIT STRING shall be the concatenation of the bitstrings[4] resulting from the encoding of the presentation data values contained in that PDV-list value according to the appropriate transfer syntax.

8.4.2.6 The presentation-context-identifier component of a PDV-list value in a CP PPDU shall identify the presentation context of the presentation data values.

8.4.2.7 The transfer-syntax-name component of a PDV-list value in a CP PPDU shall be present when more than one transfer syntax name was proposed for the presentation context of the presentation-data-values.

8.4.3 *Encoding of presentation data values in X.410-1984 mode*

8.4.3.1 Except for the S-DATA request and indication service primitives, presentation data values in type User-data shall be encoded according to Basic Encoding Rules for ASN.1 (Recommendation X.209).

[4] If the transfer syntax is not self-delimiting, then there is a danger that concatenated presentation data values will be ambiguous.

8.4.3.2 For the S-DATA request and indication service primitives, presentation data values in type User-data shall be encoded as the contents octets (i.e. no identifier octets and no length octets) of the primitive encoding of a value of type OCTET STRING, according to the Basic Encoding Rules for ASN.1 (Recommendation X.209).

8.5 *Rules of extensibility for normal mode*

8.5.1 For the CP-PPDU, a receiving PPM shall:

a) ignore any undefined element;

b) where named bits are used in § 8.2, treat any bit as insignificant when no name is assigned to it.

8.5.2 Except as specified in § 8.5.1, where named numbers or named bits are used in § 8.2, presence of a number or bit shall be invalid when no name is assigned to it.

9 Conformance

9.1 *Dynamic Conformance*

A system claiming conformance to this Recommendation shall exhibit external behaviour consistent with having implemented:

a) a PPM as defined by Section 6 and Annex A of this Recommendation;

b) use of the session-service as defined by Section 7 of this Recommendation;

c) encoding of PPDUs as defined by Section 8 of this Recommendation.

9.2 *Static Conformance*

A system claiming conformance to this Recommendation shall be capable of:

a) supporting normal mode, X.410-1984 mode, or both. A system claiming to implement the procedures specified in this Recommendation supports the procedures specified in CCITT Recommendation X.410-1984 when operating in X.410-1984 mode. A system claiming to implement the procedures specified in this Recommendation other than in support of the procedures specified in CCITT Recommendation X.410-1984 shall operate in normal mode.

b) initiating a presentation-connection (by sending a CP PPDU) or responding to a CP PPDU or both;

c) following all the remaining procedures in the presentation kernel functional unit;

d) following all the Presentation Layer procedures for each presentation functional unit that the system claims to implement and for each session functional unit which the system claims to support;

e) supporting the mapping onto the session-service defined in Section 7 of this Recommendation;

f) in normal mode, following the procedures of the rules of extensibility (§ 8.5).

9.3 *Protocol implementation conformance statement*

The Protocol Implementation Conformance Statement accompanying a system for which conformance to this Recommendation is claimed shall include statements of the following:

a) which session functional units are supported;

b) which presentation functional units are implemented;

c) whether it supports initiating a presentation-connection, responding to a CP-PPDU or both;

d) which transfer syntaxes it supports;

e) whether there is any resource-dependent limit which can cause provider rejection of a service primitive and if so, how this limit is specified;

f) whether it supports normal mode, X.410-1984 mode or both.

ANNEX A

(to Recommendation X.226)

State tables

A.1. *General*

This annex describes the presentation protocol in terms of state tables. The state tables show the state of a presentation-connection, the events that occur in the protocol, the action taken and the resultant state.

These State Tables do not constitute a formal definition of the presentation protocol; they are included to provide a more precise specification of the elements of procedure described in Section 6.

Table A-1/X.226 specifies the abbreviated name, category and name of each incoming event. The categories are PS-user event, SS-provider event and valid PPDU event.

Table A-2/X.226 specifies the abbreviated name and name of each state.

Table A-3/X.226 specifies the abbreviated name, category and name of each outgoing event. The categories are PS-provider event, SS-user event and PPDU event.

Table A-4/X.226 specifies the specific actions.

Table A-5/X.226 specifies the predicates.

Tables A-6/X.226 to A-14/X.226 specify the state tables.

A.2 *Notation for State Tables*

A.2.1 Incoming events, states and outgoing events are represented by their abbreviated names.

A.2.2 Specific actions are represented by the notation [*n*], where *n* is the number of the specific action in Table A-4/X.226.

A.2.3 Predicates are represented by the notation *pnn*, where *nn* is the number of the predicate in Table A-5/X.226.

A.2.4 Boolean operators are represented by the following notation:

&	AND
^	NOT
OR	OR

A.3 *Conventions for entries in State Tables*

A.3.1 The intersection of each state and incoming event which is invalid is left blank.

A.3.2 The intersection of each state and incoming event which is valid contains entries which are either

a) an action list which

i) may contain outgoing events and/or specific actions;

ii) always contains the resultant state;

or

b) one or more conditional action lists, each consist of

i) a predicate expression comprising predicates and boolean operators;

ii) an action list (as in § A.3.2 a)).

Note – The action lists and conditional action lists use the notation in § A.2.

A.4 *Actions to be taken by the PPM*

The state tables define the actions to be taken by the PPM.

A.4.1 *Invalid intersections*

If the intersection of the state and an incoming event is invalid, one of the following actions shall be taken.

A.4.1.1 If the incoming event comes from the PS-user, any action taken by the PPM is a local matter.

Note – One reason for the request or response service primitive being invalid is that the resulting SS-user data parameter exceeds a length limit imposed by the underlying session-service. This occurrence and its resolution are local matters.

A.4.1.2 If the incoming event is related to a received PPDU or SS-provider event, the PPM shall issue an ARP PPDU (if there is an underlying session-connection) and a P-P-ABORT indication.

A.4.2 *Valid intersections*

If the intersection of the state and incoming event is valid, one of the following actions shall be taken.

A.4.2.1 If the intersection contains an action list, the PPM shall take the specific actions in the order specified in the state table.

A.4.2.2 If the intersection contains one or more conditional action lists, for each predicate expression that is true the PPM shall take the specific actions in the order given in the action list associated with the predicate expression. If none of the predicate expressions are true, the PPM shall take one of the actions defined in § A.4.1. The order of evaluation of the predicate expressions in different conditional action lists is determined by the order of the conditional action lists.

A.4.3 *Receipt of PPDUs*

A.4.3.1 *Valid PPDUs*

The PPM shall process valid PPDUs as specified in Tables A-6/X.226 to A-14/X.226. See also § 8.5.

A.4.3.2 *Invalid PPDUs*

If an invalid PPDU is received, the PPM shall take the actions defined in § A.4.1.2.

A.5 *Definition of sets and variables*

The following sets and variables are specified.

A.5.1 *Functional units*

A set of functional units used in the procedures specified in this annex is defined as

fu-dom = (CM, CR)

where

CM = Context management functional unit

CR = Context restoration functional unit

A boolean function FU is defined over fu-dom as follows

for f in fu-dom

FU(f) = true if and only if the functional unit f has been selected during the presentation-connection establishment phase.

A.5.2 *Context sets*

In addition to the defined context set (DCS), which is implicitly used for information transfer operations, the presentation-entity needs to be aware of the following context sets:

a) presentation contexts proposed for addition, initiated locally;

b) presentation contexts proposed for addition, initiated remotely;

c) presentation contexts proposed for deletion, initiated locally;

d) presentation contexts proposed for deletion, initiated remotely;

e) the DCS agreed during presentation-connection establishment;

f) the inter-activity DCS;

g) the contents of the DCS at synchronization points.

A.5.3 *Variables*

A.5.3.1 *aep*

aep is a boolean variable having the following values:

aep = true: Activity end pending.

aep = false: Activity end not pending.

aep is set as follows:

a) aep is set true when an S-ACTIVITY-END response service primitive has been issued but while it is still possible to receive an S-ACTIVITY-INTERRUPT indication service primitive;

b) aep is set false during the presentation-connection establishment phase, or on receipt of any session-service indication primitive after an S-ACTIVITY-END response service primitive has been issued.

A.5.3.2 *rl*

rl is a boolean variable having the following values:

rl = true: Release phase started.

rl = false: Release phase not started or release has been rejected.

rl is set as follows:

a) rl is set false during the presentation-connection establishment phase or when a P-RELEASE response or confirm negative service primitive is issued;

b) rl is set true when a P-RELEASE request or indication service primitive has been issued.

A.5.3.3 *cr*

cr is a boolean variable having the following values:

cr = true: A collision of release requests is detected.

cr = false: There has not been a collision of release requests or the collision has been resolved.

cr is set as follows:

a) cr is set false during the presentation-connection establishment phase, or when rl is true and a P-RELEASE response or confirm service primitive is issued;

b) cr is set true if rl is true and a P-RELEASE request or indication service primitive is issued.

A.6 *Relationship to session-service*

In general, the behaviour of the PPM is specified independently of the behaviour of the session-service. That invocations of presentation-service primitives are acceptable to the PPM does not imply that the resulting session-service primitives will be acceptable to the session-service-provider.

Events shown in the Tables as generated by the session-service-provider or issued to the session-service-provider are implicitly conditional on the appropriate session functional unit being agreed at session-connection establishment.

TABLE A-1/X.226

Incoming Event List

Abbreviated Name	Category	Name and Description
AC	PPDU	ALTER CONTEXT
ACA	PPDU	ALTER CONTEXT ACKNOWLEDGE
ARP	PPDU	PROVIDER ABORT
ARU	PPDU	USER ABORT
CP	PPDU	PRESENTATION CONNECT
CPA	PPDU	PRESENTATION CONNECT ACCEPT
CPR	PPDU	PRESENTATION CONNECT REJECT
P-ACTDreq	PS primitive	P-ACTIVITY-DISCARD request
P-ACTDrsp	PS primitive	P-ACTIVITY-DISCARD response
P-ACTEreq	PS primitive	P-ACTIVITY-END request
P-ACTErsp	PS primitive	P-ACTIVITY-END response
P-ACTIreq	PS primitive	P-ACTIVITY-INTERRUPT request
P-ACTIrsp	PS primitive	P-ACTIVITY-INTERRUPT response
P-ACTRreq	PS primitive	P-ACTIVITY-RESUME request
P-ACTSreq	PS primitive	P-ACTIVITY-START request
P-ALTERreq	PS primitive	P-ALTER-CONTEXT request
P-ALTERrsp	PS primitive	P-ALTER-CONTEXT response
P-CDreq	PS primitive	P-CAPABILITY-DATA request
P-CDrsp	PS primitive	P-CAPABILITY-DATA response
P-CGreq	PS primitive	P-CONTROL-GIVE request
P-CONreq	PS primitive	P-CONNECT request
P-CONrsp+	PS primitive	P-CONNECT response accept
P-CONrsp–	PS primitive	P-CONNECT response reject
P-DTreq	PS primitive	P-DATA request
P-EXreq	PS primitive	P-EXPEDITED-DATA request
P-GTreq	PS primitive	P-TOKEN-GIVE request
P-PTreq	PS primitive	P-TOKEN-PLEASE request
P-RELreq	PS primitive	P-RELEASE request
P-RELrsp+	PS primitive	P-RELEASE response accept
P-RELrsp–	PS primitive	P-RELEASE response reject
P-RSYNreq	PS primitive	P-RESYNCHRONIZE request
P-RSYNrsp	PS primitive	P-RESYNCHRONIZE response
P-SYNMreq	PS primitive	P-SYNC-MAJOR request
P-SYNMrsp	PS primitive	P-SYNC-MAJOR response
P-SYNmreq	PS primitive	P-SYNC-MINOR request
P-SYNmrsp	PS primitive	P-SYNC-MINOR response
P-TDreq	PS primitive	P-TYPED-DATA request
P-UABreq	PS primitive	P-U-ABORT request
P-UERreq	PS primitive	P-U-EXCEPTION-REPORT request

TABLE A-1/X.226 *(continued)*

Incoming Event List

Abbreviated Name	Category	Name and Description
RS	PPDU	RESYNCHRONIZE
RSA	PPDU	RESYNCHRONIZE ACKNOWLEDGE
S-ACTDcnf	SS primitive	S-ACTIVITY-DISCARD confirm
S-ACTDind	SS primitive	S-ACTIVITY-DISCARD indication
S-ACTEcnf	SS primitive	S-ACTIVITY-END confirm
S-ACTEind	SS primitive	S-ACTIVITY-END indication
S-ACTIcnf	SS primitive	S-ACTIVITY-INTERRUPT confirm
S-ACTIind	SS primitive	S-ACTIVITY-INTERRUPT indication
S-ACTRind	SS primitive	S-ACTIVITY-RESUME indication
S-ACTSind	SS primitive	S-ACTIVITY-START indication
S-CGind	SS primitive	S-CONTROL-GIVE indication
S-CONcnf–	SS primitive	S-CONNECT confirm reject (provider)
S-GTind	SS primitive	S-TOKEN-GIVE indication
S-P-ABind	SS primitive	S-P-ABORT indication
S-PERind	SS primitive	S-P-EXCEPTION-REPORT indication
S-PTind	SS primitive	S-TOKEN-PLEASE indication
S-RELcnf+	SS primitive	S-RELEASE confirm accept
S-RELcnf–	SS primitive	S-RELEASE confirm reject
S-RELind	SS primitive	S-RELEASE indication
S-RSYNcnf	SS primitive	S-RESYNCHRONIZE confirm
S-RSYNind	SS primitive	S-RESYNCHRONIZE indication
S-SYNMcnf	SS primitive	S-SYNC-MAJOR confirm
S-SYNMind	SS primitive	S-SYNC-MAJOR indication
S-SYNmcnf	SS primitive	S-SYNC-MINOR confirm
S-SYNmind	SS primitive	S-SYNC-MINOR indication
S-UERind	SS primitive	S-U-EXCEPTION-REPORT indication
TC	PPDU	CAPABILITY DATA
TCC	PPDU	CAPABILITY DATA ACKNOWLEDGE
TD	PPDU	DATA
TE	PPDU	EXPEDITED DATA
TTD	PPDU	TYPED DATA

TABLE A-2/X.226

States

Abbreviated Name	Name and Description
STAI0	idle – no connection
STAI1	await CPA PPDU
STAI2	await P-CONNECT response
STAt0	connected – data transfer
STAac0	await ACA PPDU
STAac1	await P-ALTER-CONTEXT response
STAac2	await ACA PPDU or P-ALTER-CONTEXT response

TABLE A-3/X.226

Outgoing Event List

Abbreviated Name	Category	Name and Description
AC	PPDU	ALTER CONTEXT
ACA	PPDU	ALTER CONTEXT ACKNOWLEDGE
ARP	PPDU	PROVIDER ABORT
ARU	PPDU	USER ABORT
CP	PPDU	PRESENTATION CONNECT
CPA	PPDU	PRESENTATION CONNECT ACCEPT
CPR	PPDU	PRESENTATION CONNECT REJECT
P-ACTDcnf	PS primitive	P-ACTIVITY-DISCARD confirm
P-ACTDind	PS primitive	P-ACTIVITY-DISCARD indication
P-ACTEcnf	PS primitive	P-ACTIVITY-END confirm
P-ACTEind	PS primitive	P-ACTIVITY-END indication
P-ACTIcnf	PS primitive	P-ACTIVITY-INTERRUPT confirm
P-ACTIind	PS primitive	P-ACTIVITY-INTERRUPT indication
P-ACTRind	PS primitive	P-ACTIVITY-RESUME indication
P-ACTSind	PS primitive	P-ACTIVITY-START indication
P-ALTERcnf	PS primitive	P-ALTER-CONTEXT confirm
P-ALTERind	PS primitive	P-ALTER-CONTEXT indication
P-CDcnf	PS primitive	P-CAPABILITY-DATA confirm
P-CDind	PS primitive	P-CAPABILITY-DATA indication
P-CGind	PS primitive	P-CONTROL-GIVE indication
P-CONcnf+	PS primitive	P-CONNECT confirm accept
P-CONcnf−	PS primitive	P-CONNECT confirm reject
P-CONind	PS primitive	P-CONNECT indication
P-DTind	PS primitive	P-DATA indication
P-EXind	PS primitive	P-EXPEDITED-DATA indication
P-GTind	PS primitive	P-TOKEN-GIVE indication
P-PABind	PS primitive	P-P-ABORT indication
P-PERind	PS primitive	P-P-EXCEPTION-REPORT indication
P-PTind	PS primitive	P-TOKEN-PLEASE indication
P-RELcnf+	PS primitive	P-RELEASE confirm accept
P-RELcnf~	PS primitive	P-RELEASE confirm reject
P-RELind	PS primitive	P-RELEASE indication
P-RSYNcnf	PS primitive	P-RESYNCHRONIZE confirm
P-RSYNind	PS primitive	P-RESYNCHRONIZE indication
P-SYNMcnf	PS primitive	P-SYNC-MAJOR confirm
P-SYNMind	PS primitive	P-SYNC-MAJOR indication
P-SYNmcnf	PS primitive	P-SYNC-MINOR confirm
P-SYNmind	PS primitive	P-SYNC-MINOR indication
P-TDind	PS primitive	P-TYPED-DATA indication
P-UABind	PS primitive	P-U-ABORT indication
P-UERind	PS primitive	P-U-EXCEPTION-REPORT indication

TABLE A-3/X.226 *(continued)*

Outgoing Event List

Abbreviated	Category	Name and Description
RS	PPDU	RESYNCHRONIZE
RSA	PPDU	RESYNCHRONIZE acknowledge
S-ACTDreq	SS primitive	S-ACTIVITY-DISCARD request
S-ACTDrsp	SS primitive	S-ACTIVITY-DISCARD response
S-ACTEreq	SS primitive	S-ACTIVITY-END request
S-ACTErsp	SS primitive	S-ACTIVITY-END response
S-ACTIreq	SS primitive	S-ACTIVITY-INTERRUPT request
S-ACTIrsp	SS primitive	S-ACTIVITY-INTERRUPT response
S-ACTRreq	SS primitive	S-ACTIVITY-RESUME request
S-ACTSreq	SS primitive	S-ACTIVITY-START request
S-CGreq	SS primitive	S-CONTROL-GIVE request
S-GTreq	SS primitive	S-TOKEN-GIVE request
S-PTreq	SS primitive	S-TOKEN-PLEASE request
S-RELreq	SS primitive	S-RELEASE request
S-RELrsp+	SS primitive	S-RELEASE response accept
S-RELrsp−	SS primitive	S-RELEASE response reject
S-RSYNreq	SS primitive	S-RESYNCHRONIZE request
S-RSYNrsp	SS primitive	S-RESYNCHRONIZE response
S-SYNMreq	SS primitive	S-SYNCHRONIZE-MAJOR request
S-SYNMrsp	SS primitive	S-SYNCHRONIZE-MAJOR response
S-SYNmreq	SS primitive	S-SYNCHRONIZE-MINOR request
S-SYNmrsp	SS primitive	S-SYNCHRONIZE-MINOR response
S-UERreq	SS primitive	S-U-EXCEPTION-REPORT request
TC	PPDU	CAPABILITY DATA
TTC	PPDU	CAPABILITY DATA ACKNOWLEDGE
TD	PPDU	DATA
TE	PPDU	EXPEDITED DATA
TTD	PPDU	P-TYPED DATA

TABLE A-4/X.226

Specific Actions

Code	Action
[01]	Mark presentation contexts proposed for definition which provider cannot support as "provider-rejection".
[02]	Set cr and rl to FALSE.
[03]	Record abstract and transfer syntaxes for the presentation contexts of the agreed DCS and for the default context.
[04]	Propose at least one transfer syntax for each presentation context.
[05]	Propose a transfer syntax for the default context if one is named in the request service primitive.
[06]	Select one transfer syntax for each presentation context agreed for definition and include the agreed presentation contexts in the DCS.
[07]	Set rl to TRUE.
[08]	If rl is TRUE then set cr to TRUE.
[09]	If aep is TRUE then: a) set aep to FALSE; and b) if FU(CR) is TRUE then the synchronization points associated with the last activity no longer have associated DCSs.
[10]	Record selected transfer syntax for each new presentation context and include new presentation contexts in the DCS.
[11]	Remove the presentation contexts agreed for deletion from the DCS.
[12]	Record FU(f) for f in fu-dom according to the presentation requirements in the CPA PPDU.
[13]	If FU(CR) then associate the DCS with the syncpoint identifier.
[14]	If FU(CR) and an activity is in progress, set the DCS to the inter-activity DCS.
[15]	Set aep to TRUE.
[16]	Set the DCS to that associated with the syncpoint identifier.
[17]	If FU(CR) then remember the DCS as the inter-activity DCS.
[18]	Set the DCS to that agreed during presentation-connection establishment.
[19]	Eliminate any associations between syncpoint serial number and the DCS for the current activity.
[20]	Set aep to FALSE.
[21]	Set the DCS as specified by the Presentation context identifier list parameter of the PPDU.
[22]	If FU(CR), then eliminate any associations between syncpoint identifiers and DCSs.

TABLE A-5/X.226

Predicates

Code	Meaning
p01	The presentation-connection is acceptable to the PPM (local matter).
p02	If present, the named default context can be supported.
p03	Each presentation data value is from a presentation context of the DCS proposed in the presentation-connection establishment, or from the default context if this DCS is empty.
p04	Each presentation data value is from presentation contexts of the DCS being accepted in the presentation-connection establishment, or from the default context if this DCS is empty.
p05	Each presentation data value is from presentation contexts of the DCS, or from the default context if the DCS is empty.
p06	Each presentation data value is from presentation contexts of the DCS not proposed for deletion from the DCS by the peer PPM.
p07	Each presentation data value is from presentation contexts of the DCS not proposed for deletion from the DCS by the local PPM.
p08	The value of cr is TRUE.
p09	Each presentation data value if from presentation contexts of the DCS not accepted for deletion from the DCS, or from presentation contexts accepted for addition to the DCS, or if no such presentation contexts are available, from the default context.
p11	FU(CM) is TRUE.
p13	Each presentation data value is from the default context.
p14	FU(CM) is false, or FU(CM) is true and typed data functional unit was selected as a User session requirement.
p15	Each presentation data value is from presentation contexts of the DCS which was agreed during presentation-connection establishment, or from the default context if this DCS is empty.
p16	Each presentation data value is from presentation contexts of the DCS associated with the pair of Old activity identifier and Synchronization point serial number parameter values or from the default context when this DCS is empty.
p17	FU(CR) is TRUE.
p18	Each presentation date value is in presentation contexts of the DCS associated with the syncpoint identifier or from the default context if this DCS is empty.
p19	Either no syncpoint identifier is associated with a DCS or the resync identifier is not associated with a DCS and is greater than the lowest syncpoint identifier which has an associated DCS.
p20	The PPDU contains a Presentation context identifier list parameter.
p21	Each presentation data value is from presentation contexts specified in the PPDU, or from the default context if no presentation contexts are specified in the PPDU.
p22	For each presentation data value, an instance (chosen as a local matter) of encoding is supported by the PPM.
p23	For each presentation data value the encoding is supported by the PPM.
p24	Each presentation data value is from presentation contexts of the DCS, or from presentation contexts proposed for addition to the DCS by the local PPM, or from the default context if either the DCS is empty or all presentation contexts of the DCS were proposed for deletion by the local PPM.
p25	Each presentation data value is from presentation contexts of the DCS not proposed for deletion by the peer PPM or from presentation contexts proposed for addition to the DCS by the local PPM.
p26	The syncpoint identifier has an associated DCS.
p27	Old session connection identifier parameter present.
p28	There is a DCS associated with the pair of Old activity identifier and Synchronization point serial number parameters values.

TABLE A-6/X.226

Connexion Establishment

	STAI0 idle no connection	STAI1 await CPA	STAI2 await P-CONrsp
P-CONreq	p02 & p03 [04] [05] [02] [20] CP STAI1		
CP	p01 & p02 & p03 & p22 [01] [02] [20] P-CONind STAI2 ¬p01 OR ¬p02 OR ¬p22 [01] CPR STAI0		
P-CONrsp+			p04 [06] [12] CPA STAt0
CPA		p04 [03] [12] P-CONcnf+ STAt0	
P-CONrsp−			p04 [06] CPR STAI0
CPR		p04 P-CONcnf− STAI0	
S-CONcnf−		P-CONcnf− STAI0	

TABLE A-7/X.226

Connection release (normal)

	STAac0 await ACA	STAac1 await P-ALTERrsp	STAac2 await ACA or P-ALTERrsp	STAt0 connected – data transfer
P-RELreq	p07 [08] [07] S-RELreq STAac0	p05 [08] [07] S-RELreq STAac1	p07 [08] [07] S-RELreq STAac2	p05 [08] [07] S-RELreq STAt0
S-RELind	p05 [08] [07] P-RELind STAac0	p06 [08] [07] P-RELind STAac1	p06 [08] [07] P-RELind STAac2	p05 [08] [07] P-RELind STAt0
P-RELrsp+	p07 & ^p08 S-RELrsp+ STAI0 p07 & p08 [02] S-RELrsp+ STAt0	p05 & ^p08 S-RELrsp+ STAI0 p05 & p08 [02] S-RELrsp+ STAt0	p07 & ^p08 S-RELrsp+ STAI0 p07 & p08 [02] S-RELrsp+ STAt0	p05 & ^p08 S-RELrsp+ STAI0 p05 & p08 [02] S-RELrsp+ STAt0
S-RELcnf+	p05 & ^p08 P-RELcnf+ STAI0 p05 & p08 [02] P-RELcnf+ STAt0	p06 & ^p08 P-RELcnf+ STAI0 p06 & p08 [02] P-RELcnf+ STAt0	p06 & ^p08 P-RELcnf+ STAI0 p06 & p08 [02] P-RELcnf+ STAt0	p05 & ^p08 P-RELcnf+ STAI0 p05 & p08 [02] P-RELcnf+ STAt0
P-RELrsp–	p07 [02] S-RELrsp– STAac0	p05 [02] S-RELrsp– STAac1	p07 [02] S-RELrsp– STAac2	p05 [02] S-RELrsp– STAt0
S-RELcnf–	p05 [02] P-RELcnf– STAac0	p06 [02] P-RELcnf– STAac1	p06 [02] P-RELcnf– STAac2	p05 [02] P-RELcnf– STAt0

TABLE A-8/X.226

Connection release (abort)

	STAI1 await CPA	STAI2 await P-COMrsp	STAac0 await ACA	STAac1 await P-ALTERrsp	STAac2 await ACA or P-ALTERrsp	STAt0 connected – data transfer
P-UABreq	p03 ARU STAI0	p03 ARU STAI0	p07 ARU STAI0	p05 ARU STAI0	p07 ARU STAI0	p05 ARU STAI0
ARU	p03 & p21 P-UABind STAI0	p03 & p21 & p23 P-UABind STAI0	p21 & p24 P-UABind STAI0	p06 & p21 P-UABind STAI0	p21 & p25 P-UABind STAI0	p05 & p21 P-UABind STAI0
ARP	P-PABind STAI0	P-PABind STAI0	P-PABind STAI0	P-PABind STAI0	P-PABind STAI0	P-PABind STAI0
S-PABind	P-PABind STAI0	P-PABind STAI0	P-PABind STAI0	P-PABind STAI0	P-PABind STAI0	P-PABind STAI0

TABLE A-9/X.226

Context Management

	STAac0 await ACA	STAac1 await P-ALTERrsp	STAac2 await ACA or P-ALTERrsp	STAt0 connected – data transfer
P-ALTERreq		p05 [04] AC STAac2		p05 & p11 [04] AC STAac0
AC	p06 [01] P-ALTERind STAac2			p05 & p11 [01] [09] P-ALTERind STAac1
P-ALTERrsp		p09 [06] [11] ACA STAt0	p09 [06] [11] ACA STAac0	
ACA	p09 [10] [11] P-ALTERcnf STAt0		p09 & p06 [10] [11] P-ALTERcnf STAac1	

TABLE A-10/X.226

Data Transfer

	STAac0 await ACA	STAac1 await P-ALTERrsp	STAac2 await ACA or P-ALTERrsp	STAt0 connected – data transfer
P-DTreq	p07 TD STAac0	p05 TD STAac1	p07 TD STAac2	p05 TD STAt0
TD	p05 P-DTind STAac0	p06 P-DTind STAac1	p06 P-DTind STAac2	p05 [09] P-DTind STAt0
P-TDreq	p07 & p14 TTD STAac0	p05 & p14 TTD STAac1	p07 & p14 TTD STAac2	p05 & p14 TTD STAt0
TTD	p05 & p14 P-TDind STAac0	p06 & p14 P-TDind STAac1	p06 & p14 P-TDind STAac2	p05 & p14 [09] P-TDind STAt0
P-EXreq	p13 TE STAac0	p13 TE STAac1	p13 TE STAac2	p13 TE STAt0
TE	p13 P-EXind STAac0	p13 P-EXind STAac1	p13 P-EXind STAac2	p13 [09] P-EXind STAt0
P-CDreq	p07 TC STAac0	p05 TC STAac1	p07 TC STAac2	p05 TC STAt0
TC	p05 P-CDind STAac0	p06 P-CDind STAac1	p06 P-CDind STAac2	p05 [09] P-CDind STAt0
P-CDrsp	p07 TCC STAac0	p05 TCC STAac1	p07 TCC STAac2	p05 TCC STAt0
TCC	p05 P-CDcnf STAac0	p06 P-CDcnf STAac1	p06 P-CDcnf STAac2	p05 P-CDcnf STAt0

TABLE A-11/X.226

Token Handling

	STAac0 await ACA	STAac1 await P-ALTERrsp	STAac2 await ACA or P-ALTERrsp	STAt0 connected – data transfer
P-GTreq	S-GTreq STAac0	S-GTreq STAac1	S-GTreq STAac2	S-GTreq STAt0
S-GTind	P-GTind STAac0	P-GTind STAac1	P-GTind STAac2	[09] P-GTind STAt0
P-PTreq	p07 S-PTreq STAac0	p05 S-PTreq STAac1	p07 S-PTreq STAac2	p05 S-PTreq STAt0
S-PTind	p05 P-PTind STAac0	p06 P-PTind STAac1	p06 P-PTind STAac2	p05 [09] P-PTind STAt0
P-CGreq	S-CGreq STAac0	S-CGreq STAac1	S-CGreq STAac2	S-CGreq STAt0
S-CGind	P-CGind STAac0	P-CGind STAac1	P-CGind STAac2	[09] P-CGind STAt0

TABLE A-12/X.226

Synchronization

	STAac0 await ACA	STAac1 await P-ALTERrsp	STAac2 await ACA or P-ALTERrsp	STAt0 connected – data transfer
P-SYNmreq	^p17 & p07 S-SYNmreq STAac0	p05 S-SYNmreq [13] STAac1	^p17 & p07 S-SYNmreq STAac2	p05 S-SYNmreq [13] STAt0
P-SYNmind	p05 P-SYNmind [13] STAac0	^p17 & p06 P-SYNmind STAac1	^p17 & p06 P-SYNmind STAac2	p05 P-SYNmind [13] STAt0
P-SYNmrsp	p07 S-SYNmrsp STAac0	p05 S-SYNmrsp STAac1	p07 S-SYNmrsp STAac2	p05 S-SYNmrsp STAt0
S-SYNmcnf	p05 P-SYNmcnf STAac0	p06 P-SYNmcnf STAac1	p06 P-SYNmcnf STAac2	p05 P-SYNmcnf STAt0
P-SYNMreq	^p17 & p07 S-SYNMreq STAac0	p05 S-SYNMreq STAac1	^p17 & p07 S-SYNMreq STAac2	p05 S-SYNMreq STAt0
S-SYNMind	p05 P-SYNMind STAac0	^p17 & p06 P-SYNMind STAac1	^p17 & p06 P-SYNMind STAac2	p05 P-SYNMind STAt0
P-SYNMrsp	p07 S-SYNMrsp [22] [13] STAac0	^p17 & p05 S-SYNMrsp STAac1	^p17 & p07 S-SYNMrsp STAac2	p05 S-SYNmrsp [22] [13] STAt0
S-SYNcnf	^p17 & p05 P-SYNMcnf STAac0	p06 P-SYNMcnf [22] [13] STAac1	^p17 & p06 P-SYNMcnf STAac2	p05 P-SYNMcnf [22] [13] STAt0

TABLE A-13/X.226

Activity Management and Exception Handling

	STAac0 await ACA	STAac1 await P-ALTERrsp	STAac2 await ACA or P-ALTERrsp	STAt0 connected – data transfer
P-ACTSreq	^p17 & p07 S-ACTSreq STAac0	p05 [17] S-ACTSreq STAac1	^p17 & p07 S-ACTSreq STAac2	p05 [17] S-ACTSreq STAt0
S-ACTSind	p05 [09] [17] P-ACTSind STAac0	^p17 & p06 P-ACTSind STAac1	^p17 & p06 P-ACTSind STAac2	p05 [09] [17] P-ACTSind STAt0
P-ACTEreq	^p17 & p07 S-ACTEreq STAac0	p05 S-ACTEreq STAac1	^p17 & p07 S-ACTEreq STAac2	p05 S-ACTEreq STAt0
S-ACTEind	p05 P-ACTEind STAac0	^p17 & p06 P-ACTEind STAac1	^p17 & p06 P-ACTEind STAac2	p05 P-ACTEind STAt0
P-ACTErsp	p07 [14] [15] S-ACTErsp STAac0	^p17 & p05 S-ACTErsp STAac1	^p17 & p07 S-ACTErsp STAac2	p05 [14] [15] S-ACTErsp STAt0
S-ACTEcnf	^p17 & p05 P-ACTEcnf STAac0	p06 [14] [19] P-ACTEcnf STAac1	^p17 & p06 P-ACTEcnf STAac2	p05 [14] [19] P-ACTEcnf STAt0
P-ACTIreq	S-ACTIreq STAt0	S-ACTIreq STAt0	S-ACTIreq STAt0	S-ACTIreq STAt0
S-ACTIind	[20] P-ACTIind STAt0	[20] P-ACTIind STAt0	[20] P-ACTIind STAt0	[20] P-ACTIind STAt0
P-ACTIrsp				[14] S-ACTIrsp STAt0
S-ACTIcnf				[14] P-ACTIcnf STAt0
P-ACTRreq	^p17 & p07 S-ACTRreq STAac0	(^p17 OR p27 OR ^p28) & p05 S-ACTRreq STAac1 ^p27 & p28 & p17 & p16 [17] [16] S-ACTRreq STAac1	^p17 & p07 S-ACTRreq STAac2	(^p17 OR p27 OR ^p28) & p05 S-ACTRreq STAt0 ^p27 & p28 & p17 & p16 [17] [16] S-ACTRreq STAt0
S-ACTRind	(^p17 OR p27 OR ^p28) & p05 [09] P-ACTRind STAac0 ^p27 & p28 & p17 & p16 [09] [17] [16] P-ACTRind STAac0	^p17 & p06 P-ACTRind STAac1	ûp17 & p06 P-ACTRind STAac2	(^p17 OR p27 OR ^p28) & p05 [09] P-ACTRind STAt0 ^p27 & p28 & p17 & p16 [09] [17] [16] P-ACTRind STAt0

TABLE A-13/X.226 *(continued)*

Activity Management and Exception Handling

	STAac0 await ACA	STAac1 await P-ALTERrsp	STAac2 await ACA or P-ALTERrsp	STAt0 connected – data transfer
P-ACTDreq	S-ACTDreq STAt0	S-ACTDreq STAt0	S-ACTDreq STAt0	S-ACTDreq STAt0
S-ACTDind	[09] P-ACTDind STAt0	[09] P-ACTDind STAt0	[09] P-ACTDind STAt0	[09] P-ACTDind STAt0
P-ACTDrsp				[14] [19] S-ACTDrsp STAt0
S-ACTDcnf				[14] [19] P-ACTDcnf STAt0
P-UERreq	p07 S-UERreq STAt0	p05 S-UERreq STAt0	p07 S-UERreq STAt0	p05 S-UERreq STAt0
S-UERind	p05 P-UERind STAt0	p06 P-UERind STAt0	p06 P-UERind STAt0	p05 P-UERind STAt0
S-PERind	P-PERind STAt0	P-PERind STAt0	P-PERind STAt0	P-PERind STAt0

TABLE A-14/X.226

Resynchronization

	STAac0 await ACA	STAac1 await P-ALTERrsp	STAac2 await ACA or P-ALTERrsp	STAt0 connected – data transfer
P-RSYNreq				^p11 & p05 RS STAt0
	^p17 & p07 RS STAt0	^p17 & p05 RS STAt0	^p17 & p07 RS STAt0	p11 & ^p17 & p05 RS STAt0
	p17 & p19 & p07 RS STAt0	p17 & p19 & p05 RS STAt0	p17 & p19 & p07 RS STAt0	p11 & p17 & p19 & p05 RS STAt0
	p17 & p26 & p18 RS STAt0	p17 & p26 & p18 RS STAt0	p17 & p26 & p18 RS STAt0	p11 & p17 & p26 & p18 RS STAt0
	p17 & ^p19 & ^p26 & p15 [18] RS STAt0	p17 & ^p19 & ^p26 & p15 [18] RS STAt0	p17 & ^p19 & ^p26 & p15 [18] RS STAt0	p11 & p17 & ^p19 & ^p26 & p15 [18] RS STAt0
RS				^p11 & p05 P-RSYNind STAt0
	^p17 & p21 [21] P-RSYNind STAt0	^p17 & p21 P-RSYNind STAt0	^p17 & p21 [21] P-RSYNind STAt0	p11 & ^p17 & p21 P-RSYNind STAt0
	p17 & p19 & p21 [21] P-RSYNind STAt0	p17 & p19 & p21 P-RSYNind STAt0	p17 & p19 & p21 [21] P-RSYNind STAt0	p11 & p17 & p19 & p21 P-RSYNind STAt0
	p17 & p26 & p18 [16] P-RSYNind STAt0	p17 & p26 & p18 [16] P-RSYNind STAt0	p17 & p26 & p18 [16] P-RSYNind STAt0	p11 & p17 & p26 & p18 [16] P-RSYNind STAt0
	p17 & ^p19 & ^p26 & p15 [18] P-RSYNind STAt0	p17 & ^p19 & ^p26 & p15 [18] P-RSYNind STAt0	p17 & ^p19 & ^p26 & p15 [18] P-RSYNind STAt0	p11 & p17 & ^p19 & ^p26 & p15 [18] P-RSYNind STAt0
P-RSYNrsp				^p11 & p05 RSA STAt0
				p11 & ^p17 & p05 RSA STAt0
				p11 & p17 & p19 & p05 RSA STAt0
				p11 & p17 & p26 & p05 RSA STAt0
				p11 & p17 & ^p19 & ^p26 & p05 RSA STAt0

TABLE A-14/X.226 *(continued)*

Resynchronization

	STAac0 await ACA	STAac1 await P-ALTERrsp	STAac2 await ACA or P-ALTERrsp	STAt0 connected – data transfer
RSA				^p11 & p05 P-RSYNcnf STAt0 p11 & ^p17 & p21 [21] P-RSYNcnf STAt0 p11 & p17 & p19 & p21 [21] P-RSYNcnf STAt0 p11 & p17 & p26 & p05 P-RSYNcnf STAt0 p11 & p17 & ^p19 & ^p26 & p05 P-RSYNcnf STAt0

Appendix I

(to Recommendation X.226)

Differences between Recommendation X.226 and ISO International Standard 8823

I.1 Recommendation X.226 contains no statement concerning the relative precedence of any section or annex. ISO/IEC JTC 1 has indicated its intention to add a statement, prior to publication of ISO 8823, concerning precedence.

Recommendation X.227

ASSOCIATION CONTROL PROTOCOL SPECIFICATION FOR OPEN SYSTEMS INTERCONNECTION FOR CCITT APPLICATIONS[1]

(Melbourne, 1988)

The CCITT,

considering

(a) that Recommendation X.200 defines the Reference Model of Open Systems Interconnection for CCITT applications;

(b) that Recommendation X.208 specifies Abstract Syntax Notation One (ASN.1) for the specification of the abstract syntax of protocols;

(c) that Recommendation X.209 specifies the basic encoding rules for Abstract Syntax Notation One;

(d) that Recommendation X.210 defines the Open Systems Interconnection (OSI) layer service definition conventions;

(e) that Recommendation X.215 defines the Session service definition for Open Systems Interconnection for CCITT applications;

(f) that Recommendation X.216 defines the Presentation service definition of Open Systems Interconnection for CCITT applications;

(g) that Recommendation X.217 defines Association Control service definition for Open Systems Interconnection for CCITT applications;

(h) that Recommendation X.220 specifies the use of X.200 series protocols in CCITT applications;

(i) that Recommendation X.410-1984 specifies the protocol for Remote Operation and Reliable Transfer Server for Message Handling Systems; and

(j) that there is a need for common Association Control support for various applications,

unanimously declares

that this Recommendation defines the Association Control specification of Open Systems Interconnection for CCITT applications as given in the Scope and Field of Application.

CONTENTS

[1] Recommendation X.227 and ISO 8650 [Information processing systems – Open Systems Interconnection – Protocol specification for the Association Control Service Element] were developed in close collaboration and are technically aligned, except for the differences noted in Appendix I.

0 Introduction

0.1 This Recommendation is one of a set of Recommendations produced to facilitate the interconnection of information processing systems. It is related to other Recommendations in the set as defined by the Reference Model for Open Systems Interconnection (X.200). The Reference Model subdivides the area of standardization for interconnection into a series of layers of specification, each of manageable size.

0.2 The goal of Open Systems Interconnection is to allow, with a minimum of technical agreement outside the interconnection standards, the interconnection of information processing systems:

- from different manufacturers;
- under different managements;
- of different levels of complexity; and
- of different technologies.

0.3 This Recommendation specifies the protocol for the application-service-element for application-association control: the Association Control Service Element (ACSE). The ACSE provides services for establishing and releasing application-associations. These services are intended to be applicable to a wide range of application-process communication requirements.

0.4 This Recommendation includes two annexes which describe the protocol machine of ACSE in terms of a state table for normal mode of operation and for X.410-1984 mode of operation. This protocol machine is referred to as the Association Control Protocol Machine (ACPM).

0.5 The protocol defined in this Recommendation is also governed by the use of the presentation-service (X.216) and the session-service (X.215).

0.6 Quality of Services (QOS) is a parameter of the A-ASSOCIATE service. Work is still in progress to provide an integrated treatment of QOS across all of the layers of the OSI Reference Model and to ensure that the individual treatments in each layer service satisfy overall QOS objectives in a consistent manner. As a consequence, a change may be made to this Recommendation at a later time which reflects further QOS developments and integration.

1 Scope and field of application

The procedures defined in this Recommendation are applicable to instances of communication between systems which wish to interconnect in an open systems interconnection environment.

This Recommendation specifies:

a) procedures for the transfer of information relating to the application-association control between application entities; and

b) the abstract syntax for the representation of the ACSE APDUs.

The ACSE procedures are defined in terms of:

a) the interactions between peer ACSE protocol machines through the use of presentation-services; and

b) the interaction between an ACSE protocol machine and its service user.

This Recommendation also specifies conformance requirements for systems implementing these procedures. It does not contain tests which can be used to demonstrate conformance.

2 References

Recommendation X.200 – Reference Model of Open Systems Interconnection for CCITT applications (see also ISO 7498-1).

Recommendation X.208 – Specification of Abstract Syntax Notation One (see also ISO 8824).

Recommendation X.209 – Basic Encoding Rules for Abstract Syntax Notation One (see also ISO 8825).

Recommendation X.210 – OSI Layer Service Definition Conventions (see also ISO TR 8509).

Recommendation X.215 – Session service definition for Open Systems Interconnection for CCITT applications (see also ISO 8326 and ISO 8326 Addendum 2).

Recommendation X.216 – Presentation service definition for Open Systems Interconnection for CCITT applications (see also ISO 8822).

Recommendation X.217 – Association Control service definition for Open Systems Interconnection for CCITT applications (see also ISO 8649).

Recommendation X.225 – Session protocol specification for Open Systems Interconnection for CCITT applications (see also ISO 8327 and ISO 8327 Addendum 2).

Recommendation X.410 – CCITT Recommendation X.410: Message Handling Systems: Remote Operations and Reliable Transfer Server (1984).

ISO 7498-3 – Information processing systems – Open Systems Interconnection – Basic Reference Model – Part 3: Naming and Addressing.

3 Definitions

3.1 *Reference Model definitions*

This Recommendation is based on the concepts developed in X.200 and makes use of the following terms defined in it:

a) application Layer;

b) application-process;

c) application-entity;

d) application-service-element;

e) application-protocol-data-unit;

f) application-protocol-control-information;

g) presentation-service;

h) presentation-connection;

i) session-service;

j) session-service protocol; and

k) session-connection.

3.2 *Naming and addressing definitions*

This Recommendation makes use of the following terms defined in ISO 7498-3:

a) application-process title;

b) application-entity qualifier;

c) application-entity title[2]

d) application-process invocation-identifier;

e) application-entity invocation-identifier; and

f) presentation address.

3.3 *Service conventions definitions*

This Recommendation makes use of the following terms defined in X.210:

a) service-provider;

b) service-user;

c) confirmed service;

d) non-confirmed service;

e) provider-initiated service;

f) primitive;

g) request (primitive);

h) indication (primitive);

i) response (primitive); and

j) confirm (primitive).

[2] As defined in ISO 7498-3, an application-entity title is composed of an application-process title and an application-entity qualifier. The ACSE protocol provides for the transfer of an application-entity title value by the transfer of its component values.

3.4 *Presentation service definitions*

This Recommendation makes use of the following terms defined in X.216:

a) abstract syntax;

b) abstract syntax name;

c) default context;

d) defined context set;

e) functional unit [presentation];

f) normal mode [presentation];

g) presentation context;

h) presentation data value; and

i) X.410-1984 mode [presentation].

3.5 *ACSE service definitions*

This Recommendation makes use of the following terms defined in X.217.

a) application-association; association;

b) application context;

c) Association Control Service Element;

d) ACSE service-user;

e) ACSE service-provider;

f) requestor;

g) acceptor;

h) association-initiator;

i) association-responder;

j) normal mode;

k) X.410-1984 mode; and

j) disrupt.

3.6 *Association Control protocol specification definitions*

The following terms are introduced in this Recommendation.

3.6.1 **Association Control Protocol Machine**

The protocol machine for the Association Control Service Element specified in this Recommendation.

3.6.2 **requesting Association Control Protocol Machine**

The Association Control Protocol Machine whose service-user is the requestor of a particular Association Control Service Element service.

3.6.3 **accepting Association Control Protocol Machine**

The Association Control Protocol Machine whose service-user is the acceptor for a particular Association Control Service Element service.

4 **Symbols and abbreviations**

4.1 *Data units*

APDU	application-protocol-data-unit

4.2 *Types of application-protocol-data-units*

The following abbreviations have been given to the application-protocol-data-units defined in this Recommendation.

AARQ	A-ASSOCIATE-REQUEST application-protocol-data-unit
AARE	A-ASSOCIATE-REQUEST application-protocol-data-unit
RLRQ	A-RELEASE-REQUEST application-protocol-data-unit
RLRE	A-RELEASE-RESPONSE application-protocol-data-unit
ABRT	A-ABORT application-protocol-data-unit

4.3 *Other abbreviations*

The following abbreviations are used in this Recommendation:

ACPM	Association Control Protocol Machine
ACSE	Association Control Service Element
AE	application-entity
AP	application-process
APCI	application-protocol-control-information
ASE	application-service-element
ASN.1	Abstract Syntax Notation One
OSI	Open Systems Interconnection
QOS	quality of service

5 **Conventions**

5.1 This Recommendation employs a tabular presentation of its APDU fields. In § 7, tables are presented for each ACSE APDU. Each field is summarized using the following notation:

M	presence is mandatory
O	presence is ACPM option
U	presence is ACSE service-user option
req	source is related request primitive
ind	sink is related indication primitive
rsp	source is related response primitive
cnf	sink is related confirm primitive
sp	source or sink is the ACPM

5.2 The structure of each ACSE SPDU is specified in § 9 using the abstract syntax notation of ASN.1 (X.208).

6 Overview 'of the protocol

6.1 *Service provision*

The protocol specified in this Recommendation provides the services defined in X.217. These services are listed in Table 1/X.227. For a particular association, the ACSE services operate either in the normal mode or in the X.410-1984 mode. The mode of operation is determined by the Mode parameter on the A-ASSOCIATE request primitive.

TABLE 1/X.227

Service summary

Service	Type
A-ASSOCIATE	Confirmed
A-RELEASE	Confirmed
A-ABORT	Non-confirmed
A-P-ABORT	Provider-initiated

6.2 *Use of the presentation-service*

6.2.1 ACE's use of the presentation-service (X.216) is determined by ACSE's mode of operation for an association as specified below:

a) ACSE normal mode: The ACPM uses the normal mode of the presentation-service. The ACPM uses the presentation-service Kernel functional unit to exchange its APCI and, optionally, ACSE service-user information (i.e., ACSE APDUs) with its peer. The use of additional presentation-service functional units is an ACSE service-user choice. This choice does not affect the operation of the ACPM.

b) ACSE X.410-1984 mode: The ACPM uses the X.410-1984 mode of the presentation-service. Only the Kernel functional unit is available when using the presentation-service X.410-1984 mode. In this mode, the ACPM does not exchange its own APCI with its peer. It simply passes through information supplied to it by the ACSE service-user or by the presentation-service.

6.2.2 This Recommendation assumes that the ACPM is the sole user of the P-CONNECT, P-RELEASE, P-U-ABORT, and P-P-ABORT services. The ACSE neither uses nor constrains the use of any other presentation service.

6.2.3 When supported by version 1 of the session-protocol (X.225), the presentation-service is subject to length restrictions for its user-data parameters. This Recommendation assumes that a local mechanism detects violations of these constraints and makes the ACSE service-user aware of them. An encoding optimization is specified for A-ABORT to mitigate this problem (see § 7.3.3.1).

6.3 *Relationship to the session-service*

6.3.1 The functional units of the session-service (X.215) required for the session-connection which support the presentation-connection (that in turn supports the association) are determined by the A-ASSOCIATE service requestor and acceptor. They accomplish this using the Session Requirements parameter on the A-ASSOCIATE primitives.

6.3.2 The rules of the session-service affect the operation of the ACPM and its service-user. The ACSE service-user must be aware of these constraints. This Recommendation assumes that a local mechanism enforces them. Some examples of session-service constraints which affect the ACSE service-user are:

a) the availability of negotiated release; and

b) the possibility of release collisions.

6.4 *Model*

6.4.1 The Association control Protocol Machine (ACPM) is modeled as a finite state machine whose specification is given in this Recommendation. The ACPM communicates with its service-user by means of the ACSE service primitives defined in X.217. The ACPM communicates with its presentation service-provider by means of the presentation services defined in X.216.

6.4.2 The ACPM is driven by the receipt of input events from its ACSE service-user and from its presentation service-provider for the underlying presentation-connection which supports the association. The input events from the ACSE service-user are ACSE request and response primitives. The input events from its presentation service-provider are presentation indication and confirm primitives.

6.4.3 The ACPM responds to input events by issuing output events to its presentation-service-provider and to its ACSE service-user. The output events to its presentation-service-provider are presentation request and response primitives. The output events to its ACSE service-user are ACSE indication and confirm primitives.

6.4.4 The receipt of an input event, the generation of dependent actions, and the resultant output event are considered to be an indivisible action.

6.4.5 During the establishment of an association between two AEs, the existence of invocations of both the requesting and responding AEs is presumed. How they are created is outside of the scope of this Recommendation.

6.4.6 A new invocation of an ACPM is employed upon the receipt of an A-ASSOCIATE request primitive or a P-CONNECT indication primitive. Each such invocation controls exactly one association.

Note – Each association may be identified in an end system by a local mechanism so that the ACSE service-user and the ACPM can refer to the association.

6.4.7 The ACPM is modeled to operate in either one of two modes for a given association: the normal mode, and the X.410-1984 mode, as specified below.

a) When operating in the normal mode, an APCM communicates with its peer ACPM in support of an association by transferring ACSE application protocol data units (APDUs) defined in § 9[3]. An ACSE APDU is transferred as a presentation data value in the User Data parameter of the presentation primitive used on the underlying presentation-connection.

b) When operating in the X.410-1984 mode, an ACPM does not transfer ACSE APDUs with its peer. In this situation, the sending and receiving of presentation primitives are in themselves significant protocol events.

[3] This is true with one exception. If the association is supported by version 1 of the session-protocol (X.225), the requesting ACPM does not pass ACSE APCI as user data on a P-U-ABORT request primitive. The absence of ACSE APCI in this situation does not imply that the association is operating in the X.410-1984 mode (see §§ 6.4.6 and 7.3.3.1).

7 Elements of procedure

The ACSE protocol consists of the following procedures:

a) association establishment;

b) normal release of an association; and

c) abnormal release of an association.

In this clause, a summary of each of these elements of procedure is presented. This consists of a summary of the relevant APDUs, and a high-level overview of the relationship between the ACSE services, the APDUs involved, and the presentation service which is used. The use of the parameters of the presentation primitives are described in § 8.

A detailed specification of the ACSE APDUs using the ASN.1 notation (X.208) is described in § 9. Annex A specifies the state table for the ACPM for normal mode of operation. Annex B specifies the state table for the ACPM for X.410-1984 mode of operation.

7.1 *Association establishment*

7.1.1 *Purpose*

The association establishment procedure is used to establish an association between two AEs. It supports the A-ASSOCIATE service.

7.1.2 *APDUs used*

The association establishment procedure uses the A-ASSOCIATE-REQUEST (AARQ) and the A-ASSOCIATE-RESPONSE (AARE) APDUs. The fields of the AARQ PDU are listed in Table 2/X.227. The fields of the AARE APDU are listed in Table 3/X.227.

TABLE 2/X.227

AARQ APDU fields

Field name	Presence	Source	Sink
Protocol Version	O	sp	sp
Application Context Name	M	req	ind
Calling AP Title	U	req	ind
Calling AE Qualifier	U	req	ind
Calling AP Invocation-identifier	U	req	ind
Calling AE Invocation-identifier	U	req	ind
Called AP Title	U	req	ind
Called AE Qualifier	U	req	ind
Called AP Invocation-identifier	U	req	ind
Called AE Invocation-identifier	U	req	ind
Implementation information	O	sp	sp
User information	U	req	ind

TABLE 3/X.227

AARE APDU fields

Field name	Presence	Source	Sink
Protocol Version	O	sp	sp
Application Context Name	M	rsp	cnf
Responding AP Title	U	rsp	cnf
Responding AE Qualifier	U	rsp	cnf
Responding AP Invocation-identifier	U	rsp	cnf
Responding AE Invocation-identifier	U	rsp	cnf
Result	M	rsp/sp	cnf
Result Source – Diagnostic	M	rsp/sp	cnf
Implementation Information	O	sp	sp
User Information	U	rsp	cnf

7.1.3 *Association establishment procedure*

This procedure is driven by the following events:

a) an A-ASSOCIATE request primitive from the requestor;

b) an AARQ APDU as user data on a P-CONNECT indication primitive;

c) an A-ASSOCIATE response primitive from the acceptor; and

d) a P-CONNECT confirm primitive (that may or may not contain an AARE APDU).

7.1.3.1 *A-ASSOCIATE request primitive*

7.1.3.1.1 The requesting ACPM forms an AARQ APDU from parameter values of the A-ASSOCIATE request primitive and optionally, the Protocol Version and implementation information. It issues a P-CONNECT request primitive also using information from the A-ASSOCIATE request primitive. The User Data parameter of the P-CONNECT request primitive contains the AARQ APDU.

7.1.3.1.2 The requesting ACPM waits for a primitive from the presentation service-provider and does not accept any other primitive from the requestor other than an A-ABORT request primitive.

7.1.3.2 *AARQ APDU*

7.1.3.2.1 The accepting ACPM receives an AARQ APDU from its peer as user data on a P-CONNECT indication primitive.

7.1.3.2.2 The ACPM determines if the AARQ ADPU is acceptable based on the rules for extensibility (see § 7.4). If the AARQ APDU is not acceptable, a protocol error results (see § 7.3.3.4). The association establishment procedure is disrupted. An A-ASSOCIATE indication primitive is not issued. The association is not established.

7.1.3.2.3 The ACPM next inspects the value of the Protocol Version field[4] of the AARQ APDU. If the ACPM does not support a common protocol version, it forms an AARE APDU with the following assigned fields:

a) Protocol Version field (optional) with the value which indicates the protocol version(s) which it could support (see § 7.1.5.1);

b) Application Context Name field with the same value as on the AARQ APDU;

c) Result field with the value "rejected (permanent)"; and

d) Result Source-Diagnostic field with the values "ACSE service-provider" and "not common ACSE version".

In this case, the ACPM sends the AARE APDU as user data on a P-CONNECT response primitive with a Result parameter which has the value "user rejection". The ACPM does not issue an A-ASSOCIATE indication primitive. The association is not established.

7.1.3.2.4 If the P-CONNECT indication primitive and its AARQ APDU are acceptable, the ACPM issues an A-ASSOCIATE indication primitive to the acceptor. The A-ASSOCIATE indication primitive parameters are derived from the AARQ APDU and the P-CONNECT indication primitive. The ACPM waits for a primitive from the acceptor.

7.1.3.3 *A-ASSOCIATE response primitive*

7.1.3.3.1 When the accepting ACPM receives the A-ASSOCIATE response primitive, the Result parameter specifies whether the service-user has accepted or rejected the association. The ACPM forms an AARE APDU using the A-ASSOCIATE response primitive parameters. The ACPM sets the Result Source-Diagnostic field to "ACSE service-user" and the value derived from the Diagnostic parameter of the response primitive. The AARE APDU is sent as the User Data parameter on the P-CONNECT response primitive.

7.1.3.3.2 If the acceptor accepted the association resquest, the Result parameter on the related P-CONNECT response primitive specifies "acceptance", and the Result field of the outgoing AARE APDU specifies "accepted". The association is established.

7.1.3.3.3 If the acceptor rejected the association request, the Result parameter on the related P-CONNECT response primitive specifies "user-rejection", and the Result field of the AARE APDU contains the appropriate rejection value. The association is not established.

7.1.3.4 *P-CONNECT confirm primitive*

7.1.3.4.1 The requesting ACPM receives a P-CONNECT confirm primitive. The following situations are possible:

a) the association has been accepted;

b) the accepting ACPM or the acceptor has rejected the association; or

c) the representation service-provider has rejected the related presentation connection.

7.1.3.4.2 If the association was accepted, the P-CONNECT confirm primitive Result parameter specifies "acceptance". The User Data parameter contains an AARE APDU. The Result field of the AARE APDU specifies "accepted". The requesting ACPM issues an A-ASSOCIATE confirm primitive to the requestor derived from parameters from the P-CONNECT confirm primitive and the AARE APDU. The A-ASSOCIATE confirm primitive Result parameter specifies "accepted". The association is established.

7.1.3.4.3 If the association was rejected by either the accepting ACPM or by the acceptor, the related P-CONNECT confirm primitive Result parameter specifies "user-rejection". The User Data parameter contains an AARE APDU.

[4] If the Protocol Version field is not present in the AARQ APDU, version 1 is assumed

7.1.3.4.4 The requesting ACPM issues an A-ASSOCIATE confirm primitive to the requestor derived from prameters from the P-CONNECT confirm primitive and the AARE APDU. The A-ASSOCIATE confirm primitive Result parameter indicates "rejected (transient)" or "rejected (permanent)". The Result Source parameter indicates "ACSE service-user" or "ACSE service-provider". The association is not established.

7.1.3.4.5 If the presentation-connection was rejected by the presentation service-provider, the P-CONNECT confirm primitive Result parameter specifies "provider-rejection". In this situation, the User Data field is not used. The requesting ACPM issues an A-ASSOCIATE confirm primitive with the Result parameter indicating "rejected (permanent)". The Result Source parameter indicates "presentation service-provider"[5]. The association is not established.

7.1.4 *Use of the AARQ APDU fields*

The AARQ APDU fields are used by the requesting and accepting ACPMs as specified below.

7.1.4.1 *Protocol Version*

For the requesting ACPM: The value assigned to this field is determined within the implementation of the ACPM. It is a variable length bit string where each bit that is set to one indicates the version of ACSE protocol that this ACPM supports. Bit 0 represents version 1; bit 1 represents version 2; etc.. Multiple bits may be set indicating support of multiple versions. No trailing bits higher than the highest version of this Recommendation which the requesting ACPM supports are included. That is, the last bit of the string is set to one.

For the accepting ACPM: The ACPM ignores trailing bits of this field which are higher than the one indicating the latest version of this Recommendation which it supports.

7.1.4.2 *Application Context Name*

For the requesting ACPM: This value is determined by the value of the Application Context Name parameter of the A-ASSOCIATE request primitive.

For the accepting ACPM: This value is used to determine the value of the Application Context Name parameter of the A-ASSOCIATE indication primitive, if issued.

7.1.4.3 *Calling AP Title*

For the requesting ACPM: This value is determined by the value of the Calling AP Title parameter of the A-ASSOCIATE request primitive.

For the accepting ACPM: This value is used to determine the value of the Calling AP Title parameter of the A-ASSOCIATE indication primitive, if issued.

7.1.4.4 *Calling AE Qualifier*

For the requesting ACPM: This value is determined by the value of the Calling AE Qualifier parameter of the A-ASSOCIATE request primitive.

For the accepting ACPM: This value is used to determine the value of the Calling AE Qualifier parameter of the A-ASSOCIATE indication primitive, if issued.

7.1.4.5 *Calling AP Invocation-identifier*

For the requesting ACPM: This value is determined by the value of the Calling AP Invocation-identifier parameter of the A-ASSOCIATE request primitive.

For the accepting ACPM: This value is used to derive the value of the Calling AP Invocation-identifier parameter of the A-ASSOCIATE indication primitive, if issued.

[5] The presentation-service (Rec. X.216) currently does not define a Diagnostic parameter on the P-CONNECT response. However, work is still in progress to provide an integrated treatment of the "result" related parameters across all layers of the OSI Reference Model. As a consequence, a change may be made to this Recommendation at a later time that reflects further developments and integration.

7.1.4.6 *Calling AE Invocation-identifier*

For the requesting ACPM: This value is determined by the value of the Calling AE Invocation-identifier parameter of the A-ASSOCIATE request primitive.

For the accepting ACPM: This value is used to derive the value of the Calling AE Invocation-identifier parameter of the A-ASSOCIATE indication primitive, if issued.

7.1.4.7 *Called AP Title*

For the requesting ACPM: This value is determined by the value of the Called AP Title parameter of the A-ASSOCIATE request primitive.

For the accepting ACPM: This value is used to determine the value of the Called AP Title parameter of the A-ASSOCIATE indication primitive, if issued.

7.1.4.8 *Called AE Qualifier*

For the requesting ACPM: This value is determined by the value of the Called AE Qualifier parameter of the A-ASSOCIATE request primitive.

For the accepting ACPM: This value is used to determine the value of the Called AE Qualifier parameter of the A-ASSOCIATE indication primitive, if issued.

7.1.4.9 *Called AP invocation-identifier*

For the requesting ACPM: This value is determined by the value of the Called AP Invocation-identifier parameter of the A-ASSOCIATE request primitive.

For the accepting ACPM: This value is used to determine the value of the Called AP Invocation-identifier parameter of the A-ASSOCIATE indication primitive, if issued.

7.1.4.10 *Called AE Invocation-identifier*

For the requesting ACPM: This value is determined by the value of the Called AE Invocation-identifier parameter of the A-ASSOCIATE request primitive.

For the accepting ACPM: This value is determined by the value of the Called AE Invocation-identifier parameter of the A-ASSOCIATE indication primitive, if issued.

7.1.3.11 *Implementation Information*

For the requesting ACPM: The value assigned to this field is determined within the implementation of the ACPM. It contains information specific to the individual implementation of that ACPM. It is not used in negotiation.

For the accepting ACPM: This field does not affect the operation of the ACPM. Any use depends on a common understanding between the requesting and accepting ACPMs.

7.1.4.12 *User Information*

For the requesting ACPM: This value is determined by the value of the User Information parameter of the A-ASSOCIATE request primitive.

For the accepting ACPM: This value is used to determine the value of the User Information parameter of the A-ASSOCIATE indication primitive, if issued.

7.1.5 *Use of the AARE APDU fields*

The AARE APDU fields are used by the accepting and requesting ACPMs as specified below.

7.1.5.1 *Protocol Version*

For the accepting ACPM: The value of this field assigned by the ACPM depends on whether the association request is accepted or rejected by the ACPM and the acceptor as specified below.

a) If the association is accepted, the value assigned by the ACPM is a variable length bit string which indicates the protocol version selected by the ACPM from those proposed in the AARQ APDU. Only the bit indicating the version selected is set to one. That bit is the last bit in the string.

b) If the association is rejected, the value assigned by the ACPM is a variable length bit string which indicates the protocol version(s) of this Recommendation which could be supported by the ACPM.

For the requesting ACPM: The use of the value in this field depends on whether the association request is accepted or rejected.

a) If the association is accepted, this value defines the protocol version of this Recommendation to be used for this association.

b) If the association is rejected, the use of this value is a local option.

7.1.5.2 *Application Context Name*

For the accepting ACPM: This value determined by the value of the Application Context Name parameter of the A-ASSOCIATE response primitive.

For the requesting ACPM: This value is used to determine the value of the Application Context Name parameter of the A-ASSOCIATE confirm primitive.

7.1.5.3 *Responding AP Title*

For the accepting ACPM: This value is determined by the value of the Responding AP Title parameter of the A-ASSOCIATE response primitive.

For the requesting ACPM: This value is used to determine the value of the Responding AP Title parameter of the A-ASSOCIATE confirm primitive, if issued.

7.1.5.4 *Responding AE Qualifier*

For the accepting ACPM: This value is determined by the value of the Responding AE Qualifier parameter of the A-ASSOCIATE response primitive.

For the requesting ACPM: This value is used to determine the value of the Responding AE Qualifier parameter of the A-ASSOCIATE confirm primitive, if issued.

7.1.5.5 *Responding AP Invocation-Identifier*

For the accepting ACPM: This value is determined by the value of the Responding AP Invocation-identifier parameter of the A-ASSOCIATE response primitive.

For the requesting ACPM: This value is used to determine the value of the Responding AP Invocation-identifier parameter of the A-ASSOCIATE confirm primitive, if issued.

7.1.5.6 *Responding AE Invocation-identifier*

For the accepting ACPM: This value is determined by the value of the Responding AE Invocation-identifier parameter of the A-ASSOCIATE response primitive.

For the requesting ACPM: This value is used to determine the value of the Responding AE Invocation-identifier parameter of the A-ASSOCIATE confirm primitive, if issued.

7.1.5.7 *Result*

For the accepting ACPM: The value is determined by the ACPM or by the acceptor as specified below.

a) If the AARQ APDU is rejected by the ACPM (i.e., an A-ASSOCIATE indication primitive is not issued to the acceptor), the value of "rejected (permanent)" or "rejected (transient)" is assigned by the ACPM.

b) Otherwise, the value is determined by the Result parameter of the A-ASSOCIATE response primitive.

For the requesting ACPM: This value is used to determine the value of the Result parameter of the A-ASSOCIATE confirm primitive.

7.1.5.8 *Result Source-Diagnostic*

This field contains both the Result Source value and the Diagnostic value.

7.1.5.8.1 *Result Source value*

For the accepting ACPM: This value is assigned by the ACPM as specified below.

a) If the AARQ APDU is rejected by the ACPM (i.e., an A-ASSOCIATE indication primitive is not issued to the acceptor), it assigns the value "ACSE service-provider".

b) Otherwise, the ACPM assigns the value "ACSE service-user".

For the requesting ACPM: This value is used to determine the value of the Result Source parameter of the A-ASSOCIATE confirm primitive.

7.1.5.8.2 *Diagnostic value*

For the accepting ACPM: This value is determined by the ACPM or by the acceptor as specified below.

a) If the AARQ APDU is rejected by the ACPM (i.e., an A-ASSOCIATE indication primitive is not issued to the acceptor), the appropriate value is assigned by the ACPM.

b) Otherwise, the value is determined by the value of the Diagnostic parameter of the A-ASSOCIATE response primitive. If the Diagnostic parameter is not included on the response primitive, the ACPM assigns the value of "null".

For the requesting ACPM: This value is used to determine the value of the Diagnostic parameter of the A-ASSOCIATE confirm primitive, unless it has the value of "null". In this case, a Diagnostic value is not included.

7.1.5.9 *Implementation Information*

For the accepting ACPM: The value assigned to this field is determined within the implementation of the ACPM. It contains information specific to the individual implementation of that ACPM. It is not used in negotiation.

For the requesting ACPM: This field does not affect the operation of the ACPM. Any use depends on a common understanding between the accepting and requesting ACPMs.

7.1.5.10 *User Information*

For the accepting ACPM: This value is determined by the value of the User Information parameter of the A-ASSOCIATE response primitive.

For the requesting ACPM: This value is used to determine the value of the User Information parameter of the A-ASSOCIATE confirm primitive.

7.1.6 *Collisions and interactions*

7.1.6.1 *A-ASSOCIATE service*

For a given ACPM, an A-ASSOCIATE collision cannot occur (see § 6.4.6). For a given AE, two distinct ACPMs would be involved which represent the processing for two distinct associations:

a) an ACPM which processes the initial A-ASSOCIATE request primitive which results in the sending of an AARQ as user data on a P-CONNECT request primitive; and

b) an ACPM which processes the subsequently received AARQ APDU as user data on a P-CONNECT indication primitive.

7.1.6.2 *A-ABORT, P-U-ABORT, or P-P-ABORT service*

If an ACPM receives and A-ABORT request primitive, a P-U-ABORT indication primitive, or a P-P-ABORT indication primitive, it discontinues the normal association establishment procedure, and instead follows the abnormal release procedure.

7.2 *Normal release of an association*

7.2.1 *Purpose*

This procedure is used for the normal release of an association by an AE without loss of information in transit. It supports the A-RELEASE service.

7.2.2 *APDUs used*

The normal release procedure uses the A-RELEASE-REQUEST (RLRQ) APDU and the A-RELEASE-RESPONSE (RLRE) APDU. The fields of the RLRQ APDU are listed in Table 4/X.227. The fields of the RLRE APDU are listed in Table 5/X.227.

TABLE 4/X.227

RLRQ APDU fields

Field name	Presence	Source	Sink
Reason	U	req	ind
User Information	U	req	ind

TABLE 5/X.227

RLRE APDU fields

Field name	Presence	Source	Sink
Reason	U	rsp	cnf
User Information	U	rsp	cnf

7.2.3 *Normal release procedure*

This procedure is driven by the following events:

a) an A-RELEASE request primitive from the requestor;

b) an RLRQ APDU as user data on a P-RELEASE indication primitive;

c) an A-RELEASE response primitive from the acceptor, or

d) an RLRE APDU as user data on P-RELEASE confirm primitive.

7.2.3.1 *A-RELEASE request primitive*

7.2.3.1.1 When an A-RELEASE request primitive is received, the ACPM sends an RLRQ APDU as user data on a P-RELEASE request primitive using the parameters from the A-RELEASE request primitive.

Note – The requestor is required to meet the presentation (and session) requirements in order to issue an A-RELEASE request primitive (see § 6.2 and 6.3).

7.2.3.1.2 The requesting ACPM now waits for a primitive from the presentation service-provider. It does not accept any primitives from the requestor other than an A-ABORT request primitive.

7.2.3.2 *RLRQ APDU*

When the accepting ACPM receives the RLRQ APDU as user data on a P-RELEASE indication primitive, it issues an A-RELEASE indication primitive to the acceptor. It does not accept any ACSE primitives from its service-user other than an A-RELEASE response primitive or an A-ABORT request primitive.

7.2.3.3 *A-RELEASE response primitive*

The Result parameter on the A-RELEASE response primitive specifies whether the acceptor accepts or rejects the release of the association. The accepting ACPM forms an RLRE APDU from the response primitive parameters. The RLRE APDU is sent as user data on a P-RELEASE response primitive.

a) If the acceptor accepted the release, the Result parameter of the P-RELEASE response primitive has a Result parameter value of "affirmative". The association is released.

b) If the acceptor rejected the release, the Result parameter of the P-RELEASE response primitive has a Result parameter value of "negative". The association continues.

Note – To give a negative response, the acceptor is required to meet the related presentation (and session) requirements (see § 6.3).

7.2.3.4 *RLRE APDU*

The requesting ACPM receives a P-RELEASE confirm primitive containing an RLRE APDU from its peer. The Result parameter on the P-RELEASE confirm primitive specifies either that the acceptor agrees or disagrees that the association may be released. The requesting ACPM forms an A-RELEASE confirm primitive from the RLRE APDU fields.

a) If the Result parameter on the P-RELEASE confirm primitive specifies "affirmative", the association is released.

b) If the Result parameter on the P-RELEASE confirm primitive specifies "negative", the association continues. The requesting ACPM again accepts primitives from its service-user.

7.2.3.5 *A-RELEASE service collision*

7.2.3.5.1 An A-RELEASE service collision occurs when an ACPM has sent out an RLRQ APDU as the user data of a P-RELEASE request primitive (as a result of receiving an A-RELEASE request primitive from its service-user). Instead of receiving the expected RLRE APDU as uset data on a P-RELEASE confirm primitive from its peer, it receives an RLRQ APDU as the user data of a P-RELEASE indication primitive.

7.2.3.5.2 The ACPM issues an A-RELEASE indication primitive to its service-user. The procedure then followed by an ACPM depends on whether its service-user was the association-initiator or the association-responder.

a) *For the association-initiator*:

1) The ACPM waits for an A-RELEASE response primitive from its service-user. When it receives the response primitive, it forms an RLRE APDU from the response primitive's parameters. The RLRE is sent as user data on a P-RELEASE response primitive. The association continues.

2) This ACPM now waits for an RLRE from its peer as user data on a P-RELEASE confirm primitive. It does not accept any primitive from its service-user other than an A-ABORT request primitive.

3) When the ACPM receives the RLRE, it forms an A-RELEASE confirm primitive from the RLRE fields and isssues it to its service-user. The association is released.

In summary, the sequence of events which drive the ACPM of the association-initiator are:

- A-RELEASE request primitive;
- RLRQ APDU (causing the collision);
- A-RELEASE response primitive; and finally
- RLRE APDU.

b) *For the association-responder*:

1) The ACPM waits for an RLRE from its peer as user data on a P-RELEASE confirm primitive. It does not accept a primitive from its service-user other than an A-ABORT request primitive.

2) When this ACPM receives the RLRE, it forms an A-RELEASE confirm primitive from the RLRE fields. The association continues.

3) The ACPM now waits for an A-RELEASE response primitive from its service-user. When it receives the response primitive, it forms an RLRE APDU from the respone primitive's parameters. The RLRE is sent as user data on a P-RELEASE response primitive. The association is released.

In summary, the sequence of events which drive the ACPM of the association-responder are:

- A-RELEASE request primitive;
- RLRQ APDU (causing the collision);
- RLRE APDU; and finally
- A-RELEASE response primitive.

7.2.4 *Use of the RLRQ APDU fields*

The RLRQ APDU fields are used by the requesting and accepting ACPMs as specified below.

7.2.4.1 *Reason*

For the requesting ACPM: This value is determined by the value of the Reason parameter of the A-RELEASE request primitive.

For the accepting ACPM: This value is used to determine the value of the Reason parameter of the A-RELEASE indication primitive.

7.2.4.2 *User Information*

For the requesting ACPM: This value is determined by the value of the User Information parameter of the A-RELEASE request primitive.

For the accepting ACPM: This value is used to determine the value of the User Information parameter of the A-RELEASE indication primitive.

7.2.5 *Use of the RLRE APDU fields*

The RLRE APDU fields are used by the accepting and requesting ACPMs as specified below.

7.2.5.1 *Reason*

For the accepting ACPM: This value is determined by the value of the Reason parameter of the A-RELEASE response primitive.

For the requesting ACPM: This value is used to determine the value of the Reason parameter of the A-RELEASE confirm primitive.

7.2.5.2 *User Information*

For the accepting ACPM: This value is determined by the value of the User Information parameter of the A-RELEASE response primitive.

For the requesting ACPM: This value is used to determine the value of the User Information parameter of the A-RELEASE confirm primitive.

7.2.6 *Collisions and interactions*

7.2.6.1 *A-RELEASE service*

For a given ACPM, an A-RELEASE service collision can occur. The processing for such a collision is described in § 7.2.3.5.

Note – An A-RELEASE service collision can only occur if no session tokens were selected for the association.

7.2.6.2 *A-ABORT service, P-U-ABORT, or P-P-ABORT service*

If an ACPM receives an A-ABORT request primitive, a P-U-ABORT indication primitive, or a P-P-ABORT indication primitive, it disrupts the normal association release procedure, and instead follows the abnormal release procedure.

7.3 *Abnormal release of an association*

7.3.1 *Purpose*

The Abnormal Release procedure can be used at any time to force the abrupt release of the association by a requestor in either AE, by either ACPM or by the presentation service-provider. When the abnormal release procedure is applied during an attempt to establish an association, the association is not established. The abnormal release procedure supports the A-ABORT and A-P-ABORT services.

7.3.2 *APDUs used*

The abnormal release procedure uses the A-ABORT (ABRT) APDU. The fields of the ABRT APDU are listed in Table 6/X.227.

Note – No APDUs are defined for the A-P-ABORT service since it is directly mapped from the P-P-ABORT service.

TABLE 6/X.227

ABRT APDU fields

Field name	Presence	Source	Sink
Abort Source	M	sp	ind
User Information	U	req	ind

7.3.3 *Abnormal release procedure*

This procedure is driven by the following events:

a) an A-ABORT request primitive from the requestor;

b) a P-U-ABORT indication primitive;

c) a P-P-ABORT indication primitive; or

d) a protocol error detected by an ACPM.

7.3.3.1 *A-ABORT request primitive*

When an ACPM receives an A-ABORT request primitive from its service-user, the processing which it performs depends on the version of the underlying session-protocol (X.225) which supports the association as specified below.

a) For version 1, the ACPM does not send any of its APCI to its peer. It simply issues a P-U-ABORT request primitive. If the user information is included on the A-ABORT request primtive, that user information is passed as user data on the P-U-ABORT request primitive. The association is released.

b) For other versions, the ACPM sends an ABRT APDU as user data on a P-U-ABORT request primitive. The Abort Source field is specified as "ACSE service-user". If the User Information parameter is included on the A-ABORT request primitive, it is included in the ABRT APDU. The association is released.

7.3.3.2 *P-U-ABORT indication primitive*

When an ACPM receives a P-U-ABORT indication primitive, the User Data parameter may contain[6] an ABRT APDU:

a) If the indication primitive does not contain an ABRT APDU, the ACPM issues an A-ABORT indication primitive with the Abort Source parameter specified as "ACSE service-user". If a user data is contained on the P-U-ABORT indication primitive, it is included as the User Information parameter of the A-ABORT indication primitive. The association is released.

b) If the indication primitive does contain an ABRT ADPU, the ACPM issues an A-ABORT indication primitive using the Abort Source field of the ABRT APDU. If a User Information field is contained in the ABRT APDU, it is included on the A-ABORT indication primitive. The association is released.

7.3.3.3 *P-P-ABORT indication primitive*

When an ACPM receives a P-P-ABORT indication primitive, the ACPM issues an A-P-ABORT indication primitive to the acceptor. The association is released.

7.3.3.4 *Protocol errors*

7.3.3.4.1 Two types of ACSE protocol errors are possible:

a) for a particular ACPM state, an unexpected APDU is received; or

b) an invalid field is encountered during the processing of an incoming APDU (see § 7.4).

7.3.3.4.2 If an unexpected APDU is received, the abnormal release procedure is invoked. If an invalid field is detected by an ACSE procedure that procedure is disrupted and the abnormal release procedure is invoked.

[6] If an association is supported by version 1 of the session-protocol (Rec. X.225), the User Data parameter does not contain an ABRT ADPU (see § 7.3.3.1). The absence of an APDU in this situation does not imply that the application is operating in the X.410-1984 mode.

7.3.3.4.3 As part of the abnormal release procedure, the ACPM issues an A-ABORT indication primitive to its service-user, unless the error occurred during the association establishment procedure[7] as the result of receiving an invalid AARQ (see § 7.4). If an indication primitive is issued, the value of the Abort Source is "ACSE service-provider". The User Information parameter is not used.

7.3.3.4.4 The subsequent ACPM processing performed depends on the version of the underlying session-protocol (X.225) which supports the association as specified below.

a) For version 1, the ACPM issues a P-U-ABORT request primitive. No user information is included.

b) For other versions, the ACPM sends an ABRT APDU as user data on a P-U-ABORT request primitive. The Abort Source field is specified as "ACSE service-provider". The User Information field is not used.

7.3.3.4.5 In either case, the association is released.

7.3.4 *Use of the ABRT APDU fields*

The ABRT APDU fields are used by the requesting and accepting ACPMs as specified below.

7.3.4.1 *Abort Source*

For the requesting ACPM: This value is assigned by the ACPM as specified below.

a) If the ACPM initiated the abort procedure, the ACPM assigns the value of "ACSE service-provider".

b) Otherwise, the ACPM assigns the value of "ACSE service-user".

For the accepting ACPM: This value is used to determine the value of the Abort Source parameter of the A-ABORT indication primitive.

7.3.4.2 *User Information*

For the requesting ACPM: This value is determined by the value of the User Information parameter of the A-ABORT request primitive.

For the accepting ACPM: This value is used to determine the value of the User Information parameter of the A-ABORT indication primitive.

7.3.5 *Collisions and interactions*

The abnormal release procedure may be used whenever an association is established, is in the process of being established, or is being normally released. This procedure disrupts any other currently active procedure. A P-P-ABORT indication primitive can disrupt the A-ABORT procedure with loss of the A-ABORT information. Collisions of ABRT APDUs are governed by the P-U-ABORT services (X.216).

7.4 *Rules for extensibility*

7.4.1 When processing an incoming AARQ, the accepting ACPM shall:

a) ignore all tagged values which are not defined in the abstract syntax of this Recommendation; and

b) ignore all unknown bit name assignments within a bit string.

7.4.2 After the association has been established or during the establishment of an association, only those ACSE APDUs and related ADPU fields defined in the ASN.1 description of the negotiated version of this Recommendation shall be issued.

7.4.3 A received APDU or field within an APDU which is not defined in the ASN.1 description of the negotiated version of this Recommendation shall be treated as a protocol error.

[7] Since an A-ASSOCIATE indication primitive is not issued, an A-ABORT indication primitive would have no meaning, and, therefore, it is not issued.

8 Mapping to the presentation-service

This clause specifies how the presentation-service primitives are used by the ACPM. This usage depends on the mode selected (see § 6.2) for the association.

a) For the requesting ACPM: The mode for the association is determined by the value of the Mode parameter of the invoking A-ASSOCIATE request primitive. If the Mode parameter is not included on the request primitive, the default value of "normal" is used.

b) For the accepting ACPM: The mode is determined by the value of the Mode parameter of the incoming P-CONNECT indication primitive.

The usage of the presentation services for the normal mode is specified in §§ 8.1 to 8.3. The usage for the X.410-1984 mode is specified in §§ 8.4 to 8.6. Table 7/X.227 summarizes, for both modes of operation, the mapping of ACSE primitives and their related APDUs (normal mode) to the presentation primitives used.

TABLE 7/X.227

Mapping overview

ACSE primitive	APDU [a)]	Presentation Primitive
A-ASSOCIATE request/indication	AARQ	P-CONNECT request/indication
A-ASSOCIATE response/confirm	AARE	P-CONNECT response/confirm
A-RELEASE request/indication	RLRQ	P-RELEASE request/indication
A-RELEASE response/confirm	RLRE	P-RELEASE response/confirm
A-ABORT request/indication	ABRT	P-U-ABORT request/indication
A-P-ABORT indication	–	P-P-ABORT indication

[a)] ACSE APDUs are not used in the X.410-1984 mode.

8.1 *Association establishment (normal mode)*

The association establishment procedure uses the P-CONNECT service. Association establishment takes place concurrently with the establishment of the underlying presentation-connection.

8.1.1 *Directly mapped parameters*

For the P-CONNECT primitives: The following parameters are not referenced by the ACPM and are mapped directly onto the corresponding parameters of the A-ASSOCIATE primitives:

a) Calling Presentation Address;

b) Called Presentation Address;

c) Responding Presentation Address;

d) Presentation Context Definition List;

e) Presentation Context Definition Result List;

f) Default [Presentation] Context Name;

g) Default [Presentation] Context Result;

h) Quality of Service;

i) Presentation Requirement;

j) Session Requirements;

k) Initial Synchronization Point Serial Number;

l) Initial Assignment of Tokens;

m) Session-connection Identifier.

8.1.2 *Use of other P-CONNECT request and indication parameters*

The Mode and User Data parameters of the P-CONNECT request and indication primitives are referenced by the ACPM.

8.1.2.1 *Mode*

8.1.2.1.1 For the P-CONNECT request primitives: The Mode parameter is set to the value of the Mode parameter of the A-ASSOCIATE request primitive. For the normal mode of ACSE operation, this parameter has the value of "normal". This indicates to the presentation-service that it is to operate in the normal mode for this presentation-connection.

8.1.2.1.2 For the P-CONNECT indication primitive: This parameter has the value of "normal" for the normal mode of ACSE operation. The value indicates that the accepting ACPM is to operate in the normal mode for this association. The Mode parameter of the A-ASSOCIATE indication primitive is set to the value of "normal".

8.1.2.2 *User data*

For both the P-CONNECT request and indication primitives: The User Data parameter is used to carry the AARQ APDU as specified below.

a) The APCI of the AARQ APDU is expressed using the ACSE abstract syntax of this Recommendation. This abstract syntax must be included as the value of a presentation context definition parameter specified by the requestor on the A-ASSOCIATE request primitive.

   Note – The requesting and accepting ACPMs are aware of the presentation context which contains their abstract syntax by a local mechanism.

b) User information (if any) from the A-ASSOCIATE request primitive is included in the AARQ APDU and is expressed using one or more presentation contexts specified by the requestor on the A-ASSOCIATE request primitive.

8.1.3 *Use of other P-CONNECT response and confirm parameters*

The User Data and Result parameters of the P-CONNECT response and confirm primitive are referenced by the ACPM.

8.1.3.1 *Result*[8]

8.1.3.1.1 For the P-CONNECT response primitive: The Result parameter is set by the accepting ACPM as specified below.

a) If the accepting ACPM itself rejects the association, it is set as "user-rejection".

b) If the accepting ACPM accepts the request, the values is set as "acceptance", or "user-rejection" as determined by the value of the corresponding Result parameter on the A-ASSOCIATE response primitive.

[8] The AARE APDU also has a result field which must correspond to the value of this presentation parameter. The Result parameter of the A-ASSOCIATE confirm primitive is determined by the Result field of the AARE APDU.

8.1.3.1.2 For the P-CONNECT confirm primitive: The Result parameter is used by the requesting ACPM to determine if the P-CONNECT confirm primitive User Data parameter contains an AARE APDU as specified below.

a) If the Result parameter has the value "provider-rejection", the request is rejected by the presentation service-provider. The intended accepting ACPM never received the AARQ APDU. The User Data parameter does not contain an AARE APDU.

b) Otherwise, the Result parameter has the value of "acceptance" or "user rejection". The accepting ACPM received the AARQ APDU and has returned an AARE APDU which is contained in the user data parameter.

8.1.3.2 *User data*

8.1.3.2.1 The User Data field only has relevance if the presentation-connection is not rejected by the presentation service-provider (see § 8.1.3.1).

8.1.3.2.2 For both the P-CONNECT response and confirm primitives: The User Data parameter is used to carry the AARE APDU as specified below.

a) The APCI of the AARE APDU is expressed using the ACSE abstract syntax of this Recommendation. This abstract syntax must be included as the value of presentation context definition parameter selected by the acceptor on the A-ASSOCIATE response primitive.

b) User information (if any) from the A-ASSOCIATE response primitive is included in the AARE APDU and is expressed using one or more presentation contexts selected by the acceptor on the A-ASSOCIATE response primitive.

8.2 *Normal release of an association (normal mode)*

The normal release procedure uses the P-RELEASE service. The normal release of an association takes place simultaneously with the normal release of the underlying presentation-connection.

8.2.1 *Use of P-RELEASE request and indication parameters*

The User Data parameter of the P-RELEASE request and indication primitives is referenced by the ACPM.

For both the P-RELEASE request and indication primitives: The User Data parameter is used to carry the RLRQ APDU as specified below.

a) The APCI of the RLRQ APDU is expressed using the ACSE abstract syntax of this Recommendation. This abstract syntax must be one of the available presentation contexts.

b) User information (if any) from the A-RELEASE request primitive is included in the RLRQ APDU and is expressed using one or more available presentation contexts.

8.2.2 *Use of P-RELEASE response and confirm parameters*

The Result and User Data parameters of the P-RELEASE response and confirm primitives are referenced by the ACPM.

8.2.2.1 *Result*

8.2.2.1.1 For the P-RELEASE response primitive: The Result parameter is set to the value of the Result parameter of the A-RELEASE response primitive (i.e., "affirmative" or "negative"). This value indicates to the presentation service-provider whether the underlying presentation-connection is to be released or if it is to be continued.

8.2.2.1.2 For the P-RELEASE confirm primitive: The value of the Result parameter on the A-ASSOCIATE confirm primitive is set to the value of the Result parameter. This value indicates to the requesting ACPM whether the association is released or if it continues.

8.2.2.2 *User Data*

For both the P-RELEASE response and confirm primitives: The User Data parameter is used to carry the RLRE APDU as specified below.

a) The APCI of the RLRE APDU is expressed using the ACSE abstract syntax of this Recommendation. This abstract syntax must be one of the available presentation contexts.

b) User information (if any) from the A-RELEASE response primitive is included in the RLRE APDU and is expressed using one or more available presentation contexts.

8.3 *Abnormal release of an association (normal mode)*

The abnormal release procedure uses the P-U-ABORT and P-P-ABORT services. The abnormal release of an association takes place simultaneously with the abnormal release of the underlying presentation-connection.

8.3.1 *Use of P-U-ABORT request and indication parameters*

The User Data parameter of the P-U-ABORT request and indication primitives is referenced[9] by the ACPM.

For both the P-U-ABORT request and indication primitives: The User Data parameter is used to carry the ABRT APDU as specified below.

a) The APCI of the APDU is expressed using the ACSE abstract syntax of this Recommendation. This abstract syntax must be one of the available presentation contexts.

b) User information (if any) from the A-ABORT request primitive is expressed using one or more available presentation contexts.

8.3.2 *Use of P-P-ABORT indication parameter*

The reason parameter of the provider-initiated P-P-ABORT indication primitive is mapped directly to the corresponding parameter of the A-P-ABORT indication.

8.4 *Association establishment (X.410-1984 mode)*

The association establishment procedure uses the P-CONNECT service.

8.4.1 *Directly mapped parameters*

The following parameters are not referenced by the ACPM and are mapped directly onto corresponding parameters of the A-ASSOCIATE primitives:

a) User data[10];

b) Calling Presentation Address;

c) Called Presentation Address;

d) Responding Presentation Address;

e) Quality of Service;

f) Session Requirements;

g) Initial Synchronization Point Serial Number;

h) Initial Assignment of Tokens;

i) Session-connection identifier.

[9] If an association is supported by version 1 of the session-protocol (X.225), the User Data parameter is not referenced by the ACPM (because of length constraints) and is mapped directly onto the User Information parameter of the A-ABORT primitives (see § 7.3.3.1).

[10] User Data is mapped directly onto the A-ASSOCIATE User Information parameter. No explicit presentation context is available for it.

8.4.2 *Use of other P-CONNECT request and indication parameters*

The Mode parameter of the P-CONNECT request and indication primitives is referenced by the ACPM.

For the P-CONNECT request primitive: The Mode parameter is set to the value of the Mode parameter of the A-ASSOCIATE request primitive. For the X.410-1984 mode of ACSE operation, this parameter has the value of "X.410-1984". This indicates to the presentation-service that it is to operate in the X.410-1984 mode for this presentation-connection.

For the P-CONNECT indication primitive: This parameter has the value of "X.410-1984" for the X.410-1984 mode of ACSE operation. This value indicates that the accepting ACPM is to operate in the X.410-1984 mode for this association. The Mode parameter of the A-ASSOCIATE indication primitive is set to the value of "X.410-1984".

8.4.3 *Use of other P-CONNECT response and confirm parameters*

The Result parameter of the P-CONNECT response and confirm primitives is used by the ACPM when operating in the X.410-1984 mode.

For the P-CONNECT response primitive: The value of the Result parameter is mapped from the Result parameter of the A-ASSOCIATE Result parameter as shown in Table 8/X.227.

TABLE 8/X.227

Mapping ACSE Result Parameter

A-ASSOCIATE's Result	P-CONNECT's Result
accepted	acceptance
rejected (permanent)	user-rejection
rejected (transient)	user-rejection

For the P-CONNECT confirm primitive: The Result and Result source parameters of the A-ASSOCIATE confirm primitive are mapped from the Result parameter as shown in Table 9/X.227.

TABLE 9/X.227

Mapping Presentation Result Parameter

P-CONNECT's Result	A-ASSOCIATE's Result	A-ASSOCIATE's Result Source
acceptance	accepted	ACSE service-user
user-rejection	rejected (permanent)	ACSE service-user
provider-rejection	rejected (permanent)	presentation service-provider

8.5 *Normal release of an association (X.410-1984 mode)*

The normal release procedure uses the P-RELEASE service. The following parameters are not referenced by the ACPM and are mapped directly onto corresponding parameters of the A-RELEASE primitives:

a) Result;

b) User Data.

8.6 *Abnormal release of an association (X.410-1984 mode)*

The abnormal release procedure uses the P-U-ABORT and P-P-ABORT services.

8.6.1 *Use of P-U-ABORT request and indication parameters*

For both the P-U-ABORT request and indication primitives: The User Data parameter is not referenced by the ACPM and is mapped directly onto the User Information parameter of the corresponding A-ABORT primitives.

8.6.2 *Use of P-P-ABORT indication parameter*

For the P-P-ABORT indication primitive: The Reason parameter is not referenced by the ACPM and is mapped directly onto the corresponding parameter of the A-P-ABORT indication primitive.

9 Structure and encoding of ACSE APDUs

9.1 The abstract syntax of each of the ACSE APDUs is specified in this section using ASN.1 (Recommendation X.208).

```
ACSE-1 DEFINITIONS :: =

BEGIN
-- ACSE-1 refers to ACSE version 1

ACSE-apdu :: = CHOICE
    { aarq   AARQ-apdu,
      aare   AARE-apdu,
      rlrq   RLRQ-apdu,
      rlre   RLRE-apdu,
      abrt   ABRT-apdu
    }

AARQ-apdu :: = [ APPLICATION 0 ]                    IMPLICIT SEQUENCE
    { protocol-version                      [0]     IMPLICIT BIT STRING
                                                      { version1 (0) }          DEFAULT { version1 },
      application-context-name              [1]     Application-context-name
      called-AP-title                       [2]     AP-title                            OPTIONAL,
      called-AE-qualifier                   [3]     AE-qualifier                        OPTIONAL,
      called-AP-invocation-identifier       [4]     AP-invocation-identifier            OPTIONAL,
      called-AE-invocation-identifier       [5]     AE-invocation-identifier            OPTIONAL,
      calling-AP-title                      [6]     AP-title                            OPTIONAL,
      calling-AE-qualifier                  [7]     AE-qualifier                        OPTIONAL,
      calling-AP-invocation-identifier      [8]     AP-invocation-identifier            OPTIONAL,
      calling-AE-invocation-identifier      [9]     AE-invocation-identifier            OPTIONAL,
      implementation-information            [29]    IMPLICIT Implementation-data        OPTIONAL,
      user-information                      [30]    IMPLICIT Association-information     OPTIONAL
    }
```

```
AARE-apdu :: = [ APPLICATION 1 ]                               IMPLICIT SEQUENCE
     { protocol-version                              [0]       IMPLICIT BIT STRING
                                                                 { version1 (0) }            DEFAULT { version1 },
       application-context-name                      [1]       Application-context-name
       result                                        [2]       Associate-result,
       result-source-diagnostic                      [3]       Associate-source-diagnostic,
       responding-AP-title                           [4]       AP-title                          OPTIONAL,
       responding-AE-qualifier                       [5]       AE-qualifier                      OPTIONAL,
       responding-AP-invocation-identifier           [6]       AP-invocation-identifier          OPTIONAL,
       responding-AE-invocation-identifier           [7]       AE-invocation-identifier          OPTIONAL,
       implementation-information                    [29]      IMPLICIT Implementation-data      OPTIONAL,
       user-information                              [30]      IMPLICIT Association-information   OPTIONAL
     }

RLRQ-apdu :: = [ APPLICATION 2 ]                               IMPLICIT SEQUENCE
     { reason                                        [0]       IMPLICIT Release-request-reason   OPTIONAL,
       user-information                              [30]      IMPLICIT Association-information   OPTIONAL
     }

RLRE-apdu :: = [ APPLICATION 3 ]                               IMPLICIT SEQUENCE
     { reason                                        [0]       IMPLICIT Release-request-reason   OPTIONAL,
       user-information                              [30]      IMPLICIT Association-information   OPTIONAL
     }

ABRT-apdu :: = [ APPLICATION 4 ]                               IMPLICIT SEQUENCE
     { abort-source                                  [0]       IMPLICIT ABRT-source,
       user-information                              [30]      IMPLICIT Association-information   OPTIONAL
     }

ABRT-source :: = INTEGER
     { acse-service-user (0),
       acse-service-provider (1),
     }

Application-context-name :: = OBJECT IDENTIFIER

AP-title :: = ANY
          -- The exact definition and values used for AP-title
          -- should be chosen taking into account the ongoing work
          -- in areas of naming, Directories, and registration
          -- authority procedures for AP-titles, AE-titles and
          -- AE-Qualifiers

AE-qualifier :: = ANY
          -- The exact definition and values used for AE-qualifier
          -- should be chosen taking into account the ongoing work
          -- in areas of naming, Directories, and registration
          -- authority procedures for AP-titles, AE-titles and
          -- AE-Qualifiers
-- As defined in ISO 7498-3, an application-entity title
-- is composed of an application-process title and an
-- application-entity qualifier. The ACSE protocol provides
-- for the transfer of an application-entity title value
-- by the transfer of its component values. However, the
-- following data type is provided for reference by other
-- Recommendations that require a single syntactic structure
-- for AE-titles

AE-title :: = SEQUENCE  { AP-title,
                          AE-qualifier

                        }

AE-invocation-identifier :: = INTEGER

AP-invocation-identifier :: = INTEGER

Associate-result :: = INTEGER

    { accepted (0),
      rejected-permanent (1),
      rejected-transient (2)

    }

Associate-source-diagnostic :: = CHOICE

    { acse-service-user [1] INTEGER
      { null (0),
        no-reason-given (1),
        application-context-name-not-supported (2),
        calling-AP-title-not-recognized (3),
        calling-AP-invocation-identifier-not-recognized (4),
        calling-AE-qualifier-not-recognized (5),
        calling-AE-invocation-identifier-not-recognized (6),
        called-AP-title-not-recognized (7),
        called-AP-invocation-identifier-not-recognized (8),
        called-AE-qualifier-not-recognized (9),
        called-AE-invocation-identifier-not-recognized (10)

      }

      acse-service-provider [2] INTEGER

      { null (0),
        no-reason-given (1),
        no-common-acse-version (2)

      }

    }

Association-information :: = SEQUENCE OF EXTERNAL

Implementation-data :: = GraphicString

Release-request-reason :: = INTEGER

    { normal (0),
      urgent (1),
      user-defined (30)

    }
```

```
Release-response-reason :: = INTEGER
    {  normal (0),
       not-finished (1),
       user-defined (30)
    }
END
```

9.2 The following name, that has the ASN.1 type of OBJECT IDENTIFIER, applies to the ACSE abstract-syntax-definition specified in this section.

```
    {  joint-iso-ccitt association-control (2),
       abstract-syntax (1),
       apdus (0),
       version (1)
    }
```

9.3 The set of encoding rules named

```
    {  joint-iso-ccitt asn1 (1),
       basic-encoding (1) }
```

and specified in Recommendation X.209 is applicable to the ACSE abstract syntax definition.

10 Conformance

A system claiming to implement the procedures specified in this Recommendation shall comply with the requirements in § 10.1 through § 10.3.

Two modes of conformance are recognized:

a) normal mode; and

b) X.410-1984 mode.

The X.410-1984 mode exists to allow compatibility with message handling systems implementing the protocol specified in CCITT Recommendations X.410-1984.

10.1 *Statement requirements*

The following shall be stated by the implementor:

a) whether the system is capable of acting in the role of association-initiator, or association-responder, or both;

b) that the system supports version 1 of this protocol; and

c) whether the system implements:

   1) the normal mode of ACSE protocol;

   2) the X.410-1984 mode of ACSE protocol to support a message handling system; or

   3) both the normal mode and the X.410-1984 mode for the reason given in item 2) above.

10.2 *Static requirements*

The use of the Association Control Service Element is required for an application-entity to meet the minimum requirements for establishing and releasing communication with a peer entity.

10.2.1 *Normal mode*

If the normal mode is implemented, the system shall:

a) act as an association-initiator (by sending an AARQ APDU), or an association-acceptor (by responding properly to an AARQ APDU with an appropriate AARE APDU), or both, and

b) support (as a minimum) that encoding which results from applying the basic ASN.1 encoding rules to the ASN.1 specified in § 9 for the purpose of exchanging ACSE APCI.

10.2.2 *X.410-1984 mode*

If the X.410-1984 mode is implemented, the system shall act as an initiator, or acceptor, or both.

10.3 *Dynamic requirements*

10.3.1 *Normal mode*

If the normal mode is implemented, the system shall:

a) follow all the procedures specified in § 7 (including the rules for extensibility) and Annex A; and

b) support the mapping onto the Presentation Service defined in § 8.1 to § 8.3

10.3.2 *X.410-1984 mode*

If the X.410-1984 mode is implemented, the system shall support the direct mapping of parameters of presentation-service primitives onto the ACSE primitives as specified in § 8.4 to § 8.6 and Annex B.

ANNEX A

(to Recommendation X.227)

ACPM state table for normal mode of operation

This Annex forms an integral part of this Recommendation.

A.1 *General*

A.1.1 This annex defines a single Association Control Protocol machine (ACPM) for the normal mode of operation in terms of a state table (Table A-5/X.227). The state table shows the interrelationship between the state of an ACPM, the incoming events that occur in the protocol, the actions taken and, finally, the resultant state of the ACPM.

A.1.2 The ACPM state table does not constitute a formal definition of the ACPM. It is included to provide a more precise specification of the elements of procedure defined in § 7.

A.1.3 This annex contains the following tables.

a) Table A-1/X.227 specifies the abbreviated name, source, and name/description of each incoming event. The sources are:

   1) ACSE service user (AC-user);

   2) peer ACPM (AC-peer); and

   3) presentation service-provider (PS-provider).

b) Table A-2/X.227 specifies the abbreviated name of each state.

c) Table A-3/X.227 specifies the abbreviated name, target and name/description of each outgoing event. The targets are:

   1) ACSE service-user (AC-user); and

   2) peer ACPM (AC-peer).

d) Table A-4/X.227 specifies the predicates.

e) Table A-5/X.227 specifies the ACPM state table using the abbreviations of the above Tables.

TABLE A-1/X.227

Incoming event list for normal mode

Abbreviated Name	Source	Name and Description
A-ASCreq	AC-user	A-ASSOCIATE request primitive
A-ASCrsp+	AC-user	A-ASSOCIATE response primitive (Result = "accepted")
A-ASCrsp−	AC-user	A-ASSOCIATE response primitive (Result = "rejected (permanent)" or "rejected (transient)")
AARQ	AC-peer	A-ASSOCIATE-REQUEST APDU The AARQ is user data on a P-CONNECT indication
AARE+	AC-peer	A-ASSOCIATE-RESPONSE APDU (Result = "accepted") The AARE+ is user data on a P-CONNECT confirm primitive (Result = "acceptance")
AARE−	AC-peer	A-ASSOCIATE-RESPONSE APDU (Result = "reject (permanent)" or "rejected (transient)") The AARE− is user data on a P-CONNECT confirm primitive (Result = "user-rejection")
P-CONcnf−	PS-provider	P-CONNECT confirm primitive (Result = "provider-rejection")
A-RLSreq	AC-user	A-RELEASE request primitive
A-RLSrsp+	AC-user	A-RELEASE response primitive (Result = "affirmative")
A-RLSrsp−	AC-user	A-RELEASE response primitive (Result = "negative")
RLRQ	AC-peer	A-RELEASE-REQUEST APDU The RLRQ is user data on a P-RELEASE indication primitive
RLRE+	AC-peer	A-RELEASE-RESPONSE APDU The RLRE+ is user data on a P-RELEASE confirm primitive (Result = "affirmative")
RLRE−	AC-peer	A-RELEASE-RESPONSE APDU The RLRE− is user data on a P-RELEASE confirm primitive (Result = "negative")
A-ABRreq	AC-user	A-ABORT request primitive
ABRT [a)]	AC-peer	A-ABORT APDU The ABRT is user data on a P-U-ABORT indication primitive
P-PABind	PS-provider	P-P-ABORT indication primitive

[a)] When supported by version 1 of the session-protocol (X.225), the A-ABORT APDU has no APCI. The receipt of the P-U-ABORT indication implies its existence.

TABLE A-2/X.227

ACPM states for normal mode

Abbreviated Name	Description
STA0	idle: unassociated
STA1	awaiting AARE APDU
STA2	awaiting A-ASSOCIATE response
STA3	awaiting RLRE APDU
STA4	awaiting A-RELEASE response
STA5	associated
STA6	awaiting A-RELEASE response (association-initiator)
STA7	awaiting RLRE APDU (association-responder)

TABLE A-3/X.227

Outgoing event list for normal mode

Abbreviated Name	Target	Name and Description
A-ASCind	AC-user	A-ASSOCIATE indication primitive
A-ASCcnf+	AC-user	A-ASSOCIATE confirm primitive (Result = "accepted")
A-ASCcnf−	AC-user	A-ASSOCIATE confirm primitive (Result = "rejected (permanent)" or "rejected (transient)")
AARQ	AC-peer	A-ASSOCIATE-REQUEST APDU The AARQ is sent as user data on a P-CONNECT request primitive
AARE+	AC-peer	A-ASSOCIATE-RESPONSE APDU (Result = "accepted") The AARE+ is sent as user data on a P-CONNECT+ response primitive (Result = "acceptance")
AARE−	AC-peer	A-ASSOCIATE-RESPONSE APDU (Result = "rejected (permanent)" or "rejected (transient)") The AARE− is sent as user data on a P-CONNECT-response primitive (Result = "user-rejection")
A-RLSind	AC-user	A-RELEASE indication primitive
A-RLScnf+	AC-user	A-RELEASE confirm primitive (Result = "affirmative")
A-RLScnf−	AC-user	A-RELEASE confirm primitive (Result = "negative")
RLRQ	AC-peer	A-RELEASE-REQUEST APDU The RLRQ is sent as user data on a P-RELEASE request primitive
RLRE+	AC-peer	A-RELEASE-RESPONSE APDU The RLRE+ is sent as user data on a P-RELEASE response primitive (Result = "affirmative")
RLRE−	AC-peer	A-RELEASE-RESPONSE APDU The RLRE− is sent as user data on a P-RELEASE response primitive (Result = "negative")
A-ABRind	AC-user	A-ABORT indication primitive (Source = "ACSE service-user" or "ACSE service-provider")
ABRT [a)]	AC-peer	A-ABORT APDU (Source = "ACSE service-user" or "ACSE service-provider") The ABRT is sent as user data on a P-U-ABORT request primitive
A-PABind	AC-user	A-P-ABORT indication primitive

[a)] When supported by version 1 of the session-protocol X.225, the A-ABORT APDU has no APCI. The receipt of the subsequent P-U-ABORT indication implies its existence.

TABLE A-4/X.227

Predicates for normal mode

Code	Meaning
p1	ACPM can support requested connection
p2	ACPM originated this association

TABLE A-5/X.227

ACPM state table for normal mode

	STA0 Idle- Unassoc.	STA1 Awaiting AARE	STA2 Awaiting A-ASCrsp	STA3 Awaiting RLRE	STA4 Awaiting A-RLSrsp	STA5 Associated	STA6 Collision association initiator	STA7 Collision association responder
A-ASCreq	p1 AARQ STA1							
A-ASCrsp+			AARE+ STA5					
A-ASCrsp−			AARE− STA0					
AARQ	p1 A-ASCind STA2; ^p1: AARE− STA0							
AARE+		A-ASCcnf+ STA5						
AARE−		A-ASCcnf− STA0						
P-CONcnf−		A-ASCcnf− STA0						
A-RLSreq						RLRQ STA3		
A-RLSrsp+					RLRE+ STA0		RLRE+ STA3	
A-RLSrsp−					RLRE− STA5			
RLRQ				p2 A-RLSind STA6 ^p2 A-RLSind STA7		A-RLSind STA4		
RLRE+				A-RLScnf+ STA0				A-RLScnf+ STA4
RLRE−				A-RLScnf− STA5				
A-ABRreq		ABRT STA0	ABRT STA0	ABRT STA0	ABRT STA0	ABRT STA0	ABRT STA0	ABRT STA0
ABRT		A-ABRind STA0	A-ABRind STA0	A-ABRind STA0	A-ABRind STA0	A-ABRind STA0	A-ABRind STA0	A-ABRind STA0
P-PABind		A-PABind STA0	A-PABind STA0	A-PABind STA0	A-PABind STA0	A-PABind STA0	A-PABind STA0	A-PABind STA0

A.2 *Conventions*

A.2.1 The intersection of an incoming event (row) and a state (column) forms a cell.

A.2.2 In the state table, a blank cell represents the comibination of an incoming event and a state that is not defined for the ACPM (see § A.3.1).

A.2.3 A non-blank cell represents an incoming event and state that is defined for the ACPM. Such a cell contains one or more action lists. An action list may be either mandatory or conditional. If a cell contains a mandatory action list, it is the only action list in the cell.

A.2.4 A mandatory action list contains:

a) an outgoing event; and

b) a resultant state.

A.2.5 A conditional action list contains:

a) a predicate expression comprising predicates and Boolean operators (ˆ represents the Boolean NOT); and

b) a mandatory action list, this mandatory action list is used only if the predicate expression is true.

A.3 *Actions to be taken by the ACPM*

The ACPM state table defines the action to be taken by the ACPM in terms of an outgoing event and the resultant state of the ACPM.

A.3.1 *Invalid intersections*

Blank cells indicate an invalid intersection of an incoming event and state. If such an intersection occurs, one of the following actions is taken.

a) If the incoming event comes from the ACSE service-user, any action taken by the ACPM is a local matter.

b) If the incoming event is related to a received APDU or PS-provider event, the ACPM issues both an A-ABRind outgoing event (to its AC-user) and an ABRT outgoing event (to its peer ACPM).

A.3.2 *Valid intersections*

If the intersection of the state and incoming event is valid, one of the following actions is taken.

a) If a cell contains a mandatory action list the ACPM takes the actions specified;

b) If a cell contains one or more conditional action lists, for each predicate expression that is true, the ACPM takes the actions specified. If none of the predicate expressions are ture, the ACPM takes one of the actions defined in § A.3.1.

A.4 *Relationship to Presentation and other ASEs*

The ACPM state Table (Table A-5/X.227) only defines the interactions of the ACPM, its ACSE service-user and the presentation-services used by the ACPM.

Note – The occurrence of the other events from the presentation-service or other application-service-elements is not included in the ACPM state table because they do not affect the ACPM.

ANNEX B

(to Recommendation X.227)

ACPM state table for X.410-1984 mode of operation

B.1 *General*

This annex defines a single Association Control Protocol Machine (ACPM) for the X.410-1984 mode of operation in terms of a state table (Table B-5/X.227). The state table shows the interrelationship between the state of an ACPM, the incoming events that occur in the protocol, the actions taken and, finally, the resultant state of the ACPM.

For the X.410 mode of operation, the ACPM does not generate its own APDUs, but works transparently in a pass through mode. The state table is derived directly from the state table for normal mode by replacing:

- AARQ outgoing/incoming by P-CONNECT request/indication primitive;
- AARE outgoing/incoming by P-CONNECT response/confirmation primitive;
- RLRQ outgoing/incoming by P-RELEASE request/indication primitive;
- RLRE outgoing/incoming by P-RELEASE response/confirmation primitive; and
- ABRT outgoing/incoming by P-U-ABORT request/indication primitive.

A-RELEASE response negative, P-RELEASE confirm negative, A-RELEASE confirm negative and P-RELEASE response negative are omitted as they are not permitted to occur in X.410-1984 mode. Also the A-RELEASE collision case cannot occur in X.410-1984 mode, because only the initiator of the association may request the release of the association.

The initial state of an invocation of an ACPM is state 0 (STA0). Once state 0 has been left, and it is re-entered, the ACPM ceases to exist.

The ACPM state table does not constitute a formal definition of the ACPM for operation in the X.410-1984 mode. It is included to provide a more precise specification of the elements of procedure defined in § 7.

This annex contains the following tables.

a) Table B-1/X.227 specifies the abbreviated name, source, and name/description of each incoming event. The sources are:
   1) ACSE service user (AC-user);
   2) peer ACPM (AC-peer); and
   3) presentation service-provider (PS-provider).
b) Table B-2/X.227 specifies the abbreviated name of each state.
c) Table B-3/X.227 specifies the abbreviated name, target and name/description of each outgoing event. The targets are:
   1) ACSE service-user (AC-user); and
   2) peer ACPM (AC-peer).
d) Table B-4/X.227 specifies the predicates.
e) Table B-5/X.227 specifies the ACPM state table for the normal mode of operation using the abbreviations of the above tables.

B.2 *Conventions*

The intersection of an incoming event (row) and a state (column) forms a cell.

In the state table, a blank cell represents the combination of an incoming event and a state that is not defined for the ACPM (see § B.3.1).

A non-blank cell represents an incoming event and state that is defined for the ACPM. Such a cell contains one or more action lists. An action list may be either mandatory or conditional. If a cell contains a mandatory action list, it is the only action list in the cell.

A mandatory action list contains:

a) an outgoing event; and
b) a resultant state.

A conditional action list contains:

a) a predicate expression comprising predicates and Boolean operators (^ represents the Boolean NOT); and
b) a mandatory action list, this mandatory action list is used only if the predicate expression is true.

B.3 *Actions to be taken by the ACPM*

The ACPM state table defines the action to be taken by the ACPM in terms of an outgoing event and the resultant state of the ACPM.

B.3.1 *Invalid intersections*

Blank cells indicate an invalid intersection of an incoming event and state. If such an intersection occurs, one of the following actions is taken.

a) If the incoming event comes from the ACSE service-user, any action taken by the ACPM is a local matter.

b) If the incoming event is related to a PS-provider event, the ACPM issues both an A-ABRind outgoing event (to its AC-user) and a P-UABreq outgoing event (to its peer ACPM).

B.3.2 *Valid intersections*

If the intersection of the state and incoming event is valid, one of the following actions is taken.

a) If a cell contains a mandatory action list the ACPM takes the actions specified;

b) If a cell contains one or more conditional action lists, for each predicate expression that is true, the ACPM takes the actions specified. If none of the predicate expressions are true, the ACPM takes one of the actions defined in § B.3.1.

B.4 *Relationship to Presentation and other ASEs*

The ACPM state table (Table B-5/X.227) only defines the interactions of the ACPM, its ACSE service-user and the presentation-services used by the ACPM.

Note – The occurrence of the other events from the presentation-service or other application-service-elements is not included in the ACPM state table because they do not affect the ACPM.

TABLE B-1/X.227

Incoming event list for X.410-1984 mode

Abbreviated Name	Source	Name and description
A-ASCreq	AC-user	A-ASSOCIATE request primitive
A-ASCrsp+	AC-user	A-ASSOCIATE response primitive (Result = "accepted")
A-ASCresp−	AC-user	A-ASSOCIATE response primitive (Result = "rejected")
P-CONind	AC-peer	P-CONNECT indication
P-CONcnf+	AC-peer	P-CONNECT confirm primitive (Result = "accepted")
P-CONcnf−	AC-peer or PS-provider	P-CONNECT confirm primitive (Result = "user-rejection") (Result = "provider-rejection")
A-RLSreq	AC-user	A-RELEASE request primitive
A-RLSrsp+	AC-user	A-RELEASE response primitive (Result = "affirmative")
P-RELind	AC-peer	P-RELEASE indication primitive
P-RELcnf+	AC-peer	P-RELEASE confirm primitive (Result = "affirmative")
A-ABRreq	AC-user	A-ABORT request primitive
P-UABind	AC-peer	P-U-ABORT indication primitive
P-PABind	PS-provider	P-P-ABORT indication primitive

TABLE B-2/X.227

ACPM states for X.410-1984 mode

Abbreviated Name	Description
STA0	idle; unassociated
STA1	awaiting P-CONNECT confirm
STA2	awaiting A-ASSOCIATE response
STA3	awaiting P-RELEASE confirm
STA4	awaiting A-RELEASE response
STA5	associated

TABLE B-3/X.227

Outgoing event list for X.410-1984 mode

Abbreviated Name	Target	Name and description
A-ASCind	AC-user	A-ASSOCIATE indication primitive
A-ASCcnf+	AC-user	A-ASSOCIATE confirm primitive (Result = "accepted")
A-ASCcnf–	AC-user	A-ASSOCIATE confirm primitive (Result = "rejected")
P-CONreq	AC-peer	P-CONNECT request primitive
P-CONrsp+	AC-peer	P-CONNECT+ response primitive (Result = "user = rejected")
P-CONrsp–	AC-peer	P-CONNECT– response primitive (Result = "user-rejection")
A-RLSind	AC-user	A-RELEASE indication primitive
A-RLScnf+	AC-user	A-RELEASE confirm primitive (Result = "affirmative")
P-RELreq	AC-peer	P-RELEASE request primitive
P-RELrsp+	AC-peer	P-RELEASE response primitive (Result = "affirmative")
ABRind	AC-user	A-ABORT indication primitive (Source = "ACSE service-user" or "ACSE service-provider")
P-UABreq	AC-peer	P-U-ABORT request primitive (Source = "ACSE service-user" or ACSE service-provider")
A-PABind	AC-user	A-P-ABORT indication primitive

TABLE B-4/X.227

Predicates for X.410-1984 mode

Code	Meaning
p1	ACPM can support requested connection
p2	ACPM originated this association

TABLE B-5/X.227

ACPM state table for X.410-1984 mode

	STA0 Idle- unassoc.	STA1 Awaiting P-CONcnf	STA2 Awaiting A-ASCrsp	STA3 Awaiting P-RELcnf	STA4 Awaiting A-RLSrsp	STA5 Associated
A-ASCreq	p1 P-CONreq STA1					
A-ASCrsp+			P-CONrsp+ STA5			
A-ASCrsp−			P-CONrsp− STA0			
P-CONind	p1 A-ASCind STA2; ^p1: P-CONrsp− STA0					
P-CONcnf+		A-ASCcnf+ STA5				
P-CONcnf−		A-ASCcnf− STA0				
A-RLSreq						p2 P-RELreq STA3
A-RLSrsp+					P-RELrsp+ STA0	
P-RELind						^p2 A-RLSind STA4
P-RELcnf+				A-RLScnf+ STA0		
A-ABRreq		P-UABreq STA0	P-UABreq STA0	P-UABreq STA0	P-UABreq STA0	P-UABreq STA0
P-UABind		A-ABRind STA0	A-ABRind STA0	A-ABRind STA0	A-ABRind STA0	A-ABRind STA0
P-PABind		A-PABind STA0	A-PABind STA0	A-PABind STA0	A-PABind STA0	A-PABind STA0

APPENDIX I

(to Recommendation X.227)

Differences between Recommendation X.227 and ISO International Standard 8650

Recommendation X.227 and ISO 8650 are technically aligned with the following exceptions.

I.1 Clause 10 on Conformance of ISO 8650 differs from § 10 on Conformance in this Recommendation. The text that appears in this Recommendation was agreed in collaboration with ISO, and it is anticipated that the text in ISO 8650 will be amended in due course. The full text of the two sub-clauses in ISO 8650 that are different reads as follows:

"10.0.3 The X.410-1984 mode exists to allow claims of conformance to be made for message handling systems implementing the CCITT X.410-1984 series of Recommendations and, therefore, use the X.410-1984 mode of ACSE.

10.1 *Statement requirements*

The following shall be stated by the implementor:

a) whether the system is capable of acting the role of association-initiator, or association-responder, or both;

b) that the system supports version 1 of this protocol; and

c) whether the system implements:

   1) the normal mode of ACSE protocol;

   2) the X.410-1984 mode of ACSE protocol because it supports a message handling system implementing the CCITT X.400-1984 series of Recommendations; or

   3) both the normal mode and the X.410-1984 mode for the reason given in item 2) above."

I.2 This Recommendation contains no statement concerning the relative precedence of any Section or Annex. ISO 8650 contains a clause II which provides a definitive precedence statement.

I.3 This Recommendation contains an Annex B, which has not been included in ISO 8650. Annex B contains the ACPM state table information for use when the X.410-1984 mode is invoked.

I.4 There is no equivalent of this Appendix I in ISO 8650.

I.5 This Recommendation contains an Appendix II which has not yet been included in ISO 8650. Appendix II lists the OBJECT IDENTIFIER values assigned in Recommendations X.217 and X.227.

APPENDIX II

(to Recommendation X.227)

Summary of assigned object identifier values

This Appendix summarises the OBJECT IDENTIFIER values assigned in Recommendations X.217 and X.227.

```
{ joint-iso-ccitt association-control (2),
  abstract-syntax (1),
  apdus (0),
  version (1)
}
-- may be used to reference the abstract syntax
-- for Association Control defined in
-- Recommendation X.227, § 9.1.
```

Additionally Recommendation X.227 § 9.3 makes reference to the OBJECT IDENTIFIER value assigned in Recommendation X.209 for the basic encoding rules for ASN.1 as the means of specifying a transfer syntax for the abstract syntax defined in Recommendation X.227.

Recommendation X.228

RELIABLE TRANSFER: PROTOCOL SPECIFICATION[1]

(Melbourne, 1988)

The CCITT,

considering

(a) that Recommendation X.200 defines the Basic Reference Model of Open Systems Interconnection (OSI) for CCITT applications;

(b) that Recommendation X.210 defines the service conventions for describing the services of the OSI reference model;

(c) that Recommendation X.216 defines the Presentation Layer service;

(d) that Recommendation X.217 defines the Association Control service;

(e) that Recommendation X.218 defines the Reliable Transfer service;

(f) that there is a need for common Reliable Transfer support for various applications;

unanimously declares

that this Recommendation defines the Reliable Transfer protocol of Open Systems Interconnection for CCITT Applications as given in the Scope and Field of Application.

CONTENTS

[1] Recommendation X.228 and ISO 9066-2 [Information processing systems – Text Communication – Reliable Transfer Part 2: Protocol Specification] were developed in close collaboration and are technically aligned.

0 Introduction

This Recommendation specifies the protocol for the services provided by an application-service-element – the Reliable Transfer Service Element (RTSE) – to provide for the Reliable Transfer of Application Protocol Data Units (APDUs) between open systems. This Recommendation is one of a set of Recommendations specifying the protocols for sets of application-service-elements commonly used by a number of applications.

Reliable Transfer provides an application-independent mechanism to recover from communication and end-system failure minimizing the amount of retransmission.

This Recommendation is technically aligned with ISO 9066-2.

1 Scope and Field of Application

This Recommendation specifies the protocol (abstract syntax) and procedures for the Reliable Transfer Service Element services (Recommendation X.218). The RTSE services are provided in conjunction with the Association Control Service Element (ACSE) services (Recommendation X.217) and the ACSE protocol (Recommendation X.227), and the presentation-service (Recommendation X.216).

The RTSE procedures are defined in terms of:

a) the interactions between peer RTSE protocol machines through the use of ACSE and the presentation-service.

b) the interactions between the RTSE protocol machine and its service-user.

This Recommendation specifies conformance requirements for systems implementing these procedures.

2 References

Recommendation X.200 – Reference Model of Open Systems Interconnection for CCITT Applications (See also ISO 7498)

Recommendation X.208 – Specification of abstract syntax notation (See also ISO 8824).

Recommendation X.209 – Specification of Basic Encoding Rules for the abstract syntax notation (See also ISO 8825).

Recommendation X.210 – Open Systems Interconnection Layer Service Definition Conventions (See also ISO/TR 8509).

Recommendation X.216 – Presentation Service Definition for Open Systems Interconnection for CCITT Applications (See also ISO 8822).

Recommendation X.217 – Association Control Service Definition for CCITT Applications (See also ISO 8649).

Recommendation X.218 – Reliable Transfer: Model and Service Definition (See also ISO 9066-1)

Recommendation X.219 – Remote Operations: Model, Notation and Service Definition (See also ISO 9072-1)

Recommendation X.227 – Association Control Protocol Specification for CCITT Applications (See also ISO 8650).

3 Definitions

3.1 *Reference Model Definitions*

This Recommendation is based on the concepts developed in Recommendation X. 200 and makes use of the following terms defined in it:

a) Application Layer;

b) application-process;

c) application-entity;

d) application-service-element;
e) application-protocol-data-unit;
f) application-protocol-control-information;
g) presentation-service;
h) presentation-connection;
i) session-service;
j) session-connection; and
k) user-element
l) two-way-alternate interaction
m) transfer syntax.

3.2 *Service Conventions Definitions*

This Recommendation makes use of the following terms defined in Recommendation X.210:

a) service-provider;
b) service-user;
c) confirmed service;
d) non-confirmed service;
e) provider-initiated service;
f) primitive;
g) request (primitive);
h) indication (primitive);
i) response (primitive); and
j) confirm (primitive).

3.3 *Presentation Service Definitions*

This Recommendation makes use of the following terms defined in Recommendation X.216:

a) abstract syntax;
b) abstract syntax name;
c) presentation context;
d) default context.

3.4 *Association Control Definitions*

This Recommendation makes use of the following terms defined in Recommendation X.217:

a) application-association; association;
b) application context;
c) Association Control Service Element; and
d) X.410-1984 mode.

3.5 *RTSE Service Definitions*

This Recommendation makes use of the following terms defined in Recommendation X.218:

a) association-initiating-application-entity; association-initiator;
b) association-responding-application-entity; association-responder;
c) sending-application-entity; sender;
d) receiving-application-entity; receiver;
e) requestor;
f) acceptor;
g) Reliable Transfer Service Element;
h) RTSE-user;
i) RTSE-provider;
j) ACSE-provider;

k) monologue interaction;

l) syntax-matching-services;

m) Reliable Transfer;

n) X.410-1984 mode; and

o) normal mode.

3.6 *Reliable Transfer Protocol Specification Definitions*

For the purpose of this Recommendation the following definitions apply:

3.6.1 **reliable-transfer-protocol-machine**:

The protocol machine for the Reliable Transfer Service Element specified in this Recommendation.

3.6.2 **requesting-reliable-transfer-protocol-machine**:

The reliable-transfer-protocol-machine whose RTSE-user is the requestor of a particular Reliable Transfer Service Element service.

3.6.3 **accepting-reliable-transfer-protocol-machine**:

The reliable-transfer-protocol-machine whose RTSE-user is the acceptor for a particular Reliable Transfer Service Element service.

3.6.4 **sending-reliable-transfer-protocol-machine**:

The reliable-transfer-protocol-machine whose RTSE-user is the sender.

3.6.5 **receiving-reliable-transfer-protocol-machine**:

The reliable-transfer-protocol-machine whose RTSE-user is the receiver.

3.6.6 **association-initiating-reliable-transfer-protocol-machine**:

The reliable-transfer-protocol-machine whose RTSE-user is the association-initiator.

3.6.7 **association-responding-reliable-transfer-protocol-machine**:

The reliable-transfer-protocol-machine whose RTSE-user is the association-responder.

4 Abbreviations

4.1 *Data Units*

APDU	application-protocol-data-unit.

4.2 *Types of Application-protocol-data-units*

The following abbreviations have been given to the application-protocol-data-units defined in this Recommendation.

RTAB	RT-P-ABORT and RT-U-ABORT application-protocol-data-unit
RTORQ	RT-OPEN-REQUEST application-protocol-data-unit
RTOAC	RT-OPEN-ACCEPT application-protocol-data-unit
RTORJ	RT-OPEN-REJECT application protocol-data-unit
RTTR	RT-TRANSFER application-protocol-data-unit
RTTP	RT-TOKEN-PLEASE application-protocol-data-unit

4.3 *Other Abbreviations*

The following abbreviations are used in this Recommendation.

AE	application-entity
ACSE	Association Control Service Element
ASE	application-service-element
RTPM	reliable-transfer-protocol-machine
RT (or RTS)	Reliable Transfer
RTSE	Reliable Transfer Service Element

5 Conventions

This Recommendation employs a tabular presentation of its APDU fields. In clause 7, tables are presented for each RTSE APDU. Each field is summarized using the following notation:

M	presence is mandatory
U	presence is a RTSE service-user option
T	presence is a RTPM option
req	source is related request primitive
ind	sink is related indication primitive
resp	source is related response primitive
conf	sink is related confirm primitive
sp	source or sink is the RTPM

The structure of each RTSE APDU is specified in clause 9 using the abstract syntax notation of Recommendation X.208.

6 Overview of the Protocol

6.1 *Service Provision*

The protocol specified in this Recommendation provides the services defined in Recommendation X.218. These services are listed in Table 1/X.228.

TABLE 1/X.228

RTSE Service Summary

Service	Type
RT-OPEN	Confirmed
RT-CLOSE	Confirmed
RT-TRANSFER	Confirmed
RT-TURN-PLEASE	Non-confirmed
RT-TURN-GIVE	Non-confirmed
RT-P-ABORT	Provider-initiated
RT-U-ABORT	Non-confirmed

6.2 *Use of Services*

6.2.1 *ACSE Services*

The RTPM requires access to the A-ASSOCIATE, A-RELEASE, A-ABORT and A-P-ABORT services. This Recommendation assumes that the RTPM is the sole user of these services.

6.2.2 *Use of the Presentation-service*

The RTPM requires access to the P-ACTIVITY-START, P-DATA, P-MINOR-SYNCHRONIZE, P-ACTIVITY-END, P-ACTIVITY-INTERRUPT, P-ACTIVITY-DISCARD, P-U-EXCEPTION-REPORT, P-ACTIVITY-RESUME, P-P-EXCEPTION-REPORT, P-TOKEN-PLEASE and P-CONTROL-GIVE services. This Recommendation assumes that the RTPM is the sole user of the above services.

The RTPM requires access to local syntax-matching-services provided by the presentation-service provider. This syntax-matching-service consists of:

a) an encoding service enabling the transformation from the local representation of an APDU value into an encoded-APDU-value of type OCTET STRING the value of which is the representation of the APDU value specified by the negotiated transfer syntax;

b) a decoding service enabling the transformation from an encoded-APDU-value into the local representation of the APDU value.

If X.410-1984 mode or simple encoding is used by the Presentation Layer, the APDU value is encoded as ASN.1 type ANY. If full encoding is used by the Presentation Layer, the APDU value is encoded as ASN.1 type EXTERNAL. (For X.410-1984 mode, single encoding and full encoding see Recommendation X.226).

This Recommendation recognizes that the ACSE services require access to the P-CONNECT, P-RELEASE, P-U-ABORT and P-P-ABORT services. This Recommendation assumes that the ACSE and the RTPM are the sole users of any of the above, or of any other, presentation-services.

During the lifetime of an application-association, the underlying presentation-connections either use a single presentation context or multiple presentation contexts as a part of the presentation multiple defined context facility. The choice is determined by the use of the Single Presentation Context parameter of the RT-OPEN service as described in clause 8.1.1.1.3 and 8.1.1.1.4.

6.3 *Model*

The reliable-transfer-protocol-machine (RTPM) communicates with its service-user by means of primitives defined in Recommendation X.218. Each invocation of the RTPM controls a single application-association.

The RTPM is driven by RTSE service request and response primitives from its service-user, and by indication and confirm primitives from the ACSE services and the presentation-service. The RTPM, in turn, issues indication and confirm primitives to its service-user and request and response primitives on the used ACSE services or presentation-service.

The reception of an RTSE service primitive, or of an ACSE service primitive, or of a presentation-service primitive, and the generation of dependent actions are considered to be indivisible.

During the use of the RTSE services, the existence of both the association-initiating AE and the association-responding AE is presumed. How these AEs are created is beyond the scope of this Recommendation.

During the use of the RTSE services, except RT-OPEN, the existence of an application-association between the peer AEs is presumed.

Note – Each application-association may be identified in an end system by an internal, implementation dependent mechanism so that the RTSE service-user, and the RTPM, and the ACSE service-provider can refer to it.

7 Elements of procedure

The RTSE protocol consists of the following elements of procedure:

a) association-establishment

b) association-release

c) transfer

d) turn-please

e) turn-give

f) error reporting:

  f1) user-exception-report

  f2) provider-exception-report

g) error handling:

  g1) transfer-interrupt

  g2) transfer-discard

  g3) association-abort

  g4) association-provider-abort

h) error recovery:

  h1) transfer-resumption (for recovery from g1), or after successful h3) from g3) or g4))

  h2) transfer-retry (for recovery from g2))

  h3) association-recovery (for recovery from g3 or g4))

i) abort:

  i1) transfer-abort (recovery from g1) or g2) or g3) or g4) not possible)

  i2) provider-abort (recovery from g1) or g2) or g3) or g4) not possible)

  i3) user-abort.

In the following clauses, a summary of each of these elements of procedure is presented. This consists of a summary of the relevant APDUs, and a high-level overview of the relationship between the RTSE service primitives and the APDUs involved and the used presentation-service.

Clause 8 describes how the service primitives are mapped on the ACSE services, and on the presentation-service.

7.1 *Association-establishment*

7.1.1 *Purpose*

The association-establishment procedure is used to establish an application-association.

7.1.2 *APDUs used*

The association-establishment procedures uses the RT-OPEN-REQUEST (RTORQ) APDU, the RT-OPEN-ACCEPT (RTCAC) APDU, and the RT-OPEN-REJECT (RTORJ) APDU.

Note – These APDUs are also used in the association-recovery procedure.

7.1.2.1 *RTORQ APDU*

The RT-OPEN REQUEST (RTORQ) APDU is used in the request to establish an application-association. The fields of the RTORQ APDU are listed in Table 2/X.228.

7.1.2.2 *RTOAC APDU*

The RT-OPEN-ACCEPT (RTOAC) APDU is used in the positive response to the request to establish an application-association. The fields of the RTOAC APDU are listed in Table 3/X.228.

TABLE 2/X.228

RTORQ APDU Fields

Field name	Presence	Source	Sink
Checkpoint-size	T	sp	sp
Window-size	T	sp	sp
Dialogue-mode	U	req	ind
User-data (Note 1)	U	req	ind
Session-connection-identifier (Note 2)	T	sp	sp
Application-protocol (Note 3)	U	req	ind

Note 1 – The User-data field is used solely in the association-establishment procedure.

Note 2 – The Session-connection-identifier field is used solely in the association-recovery procedure.

Note 3 – The Application-protocol field is used solely in the X.410-1984 mode.

TABLE 3/X.228

RTOAC APDU Fields

Field name	Presence	Source	Sink
Checkpoint-size	T	sp	sp
Window-size	T	sp	sp
User-data (Note 1)	U	resp	conf
Session-connection-identifier (Note 2)	T	sp	sp

Note 1 – The User-data field is used solely in the association-establishment procedure.

Note 2 – The Session-connection-identifier field is used solely in the association-recovery procedure.

7.1.2.3 *RTORJ APDU*

RT-OPEN-REJECT (RTORJ) APDU is used in the negative response to the request to establish an application-association. The fields of the RTORJ APDU are listed in Table 4/X.228.

TABLE 4/X.228

RTORJ APDU Fields

Field name	Presence	Source	Sink
Refuse-reason (Note 1)	T	sp	sp
User-data (Note 2)	U	resp	conf

Note 1 – The Refuse-reason field is used solely in the X.410-1984 mode.

Note 2 – The User-data field is used solely in the normal mode, and is not used in the association-recovery procedure.

7.1.3 *Association-establishment procedure*

This procedure is driven by the following events:

a) an RT-OPEN request primitive from the requestor (association-initiator);

b) an RTORQ APDU as user-data on an A-ASSOCIATE indication primitive;

c) an RT-OPEN response primitive from the acceptor (association-responder);

d) an A-ASSOCIATE confirm primitive that may contain either an RTOAC APDU, or an RTORJ APDU, or no APDU.

7.1.3.1 *RT-OPEN request primitive*

The requesting RTPM forms an RTORQ APDU from the parameter values of the RT-OPEN request primitive and its internal data. The RT-OPEN request primitive parameters, except user-data, are stored by the requesting RTPM for association-recovery. The requesting RTPM issues an A-ASSOCIATE request primitive also using information from the RT-OPEN request primitive. The RTORQ APDU is the user information parameter value of the A-ASSOCIATE request primitive.

The requesting RTPM waits for a primitive from the ACSE-provider and does not accept any other primitive from the requestor.

7.1.3.2 *RTORQ APDU*

If the application-association is not accepted by the ACSE-provider, no A-ASSOCIATE indication primitive is received by the accepting RTPM and no action takes place.

If the application-association is accepted by the ACSE-provider, the accepting RTPM receives the RTORQ APDU as the user information parameter on an A-ASSOCIATE indication primitive.

If any of the A-ASSOCIATE indication parameters or any of the RTORQ APDU fields is unacceptable to the accepting RTPM, or if the accepting RTPM is not able to accept the application-association, it forms and sends an RTORJ APDU with appropriate parameters from internal data. The accepting RTPM issues an A-ASSOCIATE response primitive. The RTORJ APDU is sent as the user information parameter of the A-ASSOCIATE response primitive. The application-association is not established. The accepting RTPM does not issue an RT-OPEN indication.

If the A-ASSOCIATE indication primitive and the RTORQ APDU parameters are acceptable to the accepting RTPM, it issues an RT-OPEN indication primitive to the acceptor. The RT-OPEN indication parameter values are derived from the RTORQ APDU and the A-ASSOCIATE indication primitive parameter values.

The accepting RTPM waits for an RT-OPEN response primitive from the acceptor, or a primitive from the ACSE-provider.

7.1.3.3 *RT-OPEN response primitive*

When the accepting RTPM receives an RT-OPEN response primitive from the acceptor, the result parameter specifies whether the acceptor has accepted (value "accepted") or rejected the application-association.

If the application-association is accepted by the acceptor, the accepting RTPM forms an RTOAC APDU using the RT-OPEN response primitive parameters and internal data. The RT-OPEN response primitive parameters, except user data, are stored by the accepting RTPM for association-recovery.

The accepting RTPM issues an A-ASSOCIATE response primitive also using information from the RT-OPEN response primitive. The RTOAC APDU is sent as the user information parameter of the A-ASSOCIATE response primitive.

If the application-association is rejected by the acceptor, the accepting RTPM forms a RTORJ APDU using the RT-OPEN response primitive parameters and internal data. The accepting RTPM issues an A-ASSOCIATE response primitive also using information from the RT-OPEN request primitive. The RTORJ APDU is sent as the user information parameter of the A-ASSOCIATE response primitive. The application-association is not established.

7.1.3.4 *A-ASSOCIATE confirm primitive*

The requesting RTPM receives an A-ASSOCIATE confirm primitive. The following situations are possible:

a) the application-association has been accepted by the acceptor;

b) the accepting RTPM or the acceptor has rejected the application-association; or

c) the ACSE service provider has rejected the application-association.

If the application-association was accepted by the acceptor, the A-ASSOCIATE confirm primitive result parameter has the value "accepted" and the RTOAC APDU is the value of the user information parameter of the A-ASSOCIATE confirm primitive. The requesting RTPM issues an RT-OPEN confirm primitive to the requestor. The result parameter has the value "accepted" and the user-data parameter contains the user-data parameter value of the RTOAC APDU. The other parameters of the RT-OPEN confirm primitive are derived from the A-ASSOCIATE confirm primitive.

If the application-association was rejected by either the acceptor or the accepting RTPM, the A-ASSOCIATE confirm primitive result parameter has one of the values "rejected...", the A-ASSOCIATE confirm primitive result source parameter has the value "ACSE service-user" and the RTORJ APDU is the value of the user information parameter of the A-ASSOCIATE confirm primitive. The requesting RTPM issues an RT-OPEN confirm primitive to the requestor. The result parameter has one of the values "rejected..." and the other parameter values are derived from the A-ASSOCIATE confirm primitive parameters and the RTORJ APDU. The application-association is not established.

If the application-association was rejected by the ACSE service-provider, the A-ASSOCIATE confirm primitive result parameter has one of the values "rejected...", the A-ASSOCIATE confirm primitive result source parameter has either the value "ACSE service-provider" or "presentation service-provider". The user-data parameter of the RT-OPEN confirm primitive is absent and the application-association is not established. The other parameters of the RT-OPEN confirm primitive are derived from the A-ASSOCIATE confirm primitive.

7.1.4 *Use of the RTORQ APDU fields*

The RTORQ APDU fields are used as follows.

7.1.4.1 *Checkpoint-size*

The checkpoint-size field allows negotiation of the maximum amount of data (in units of 1024 octets) that may be sent between two minor synchronization points. A value of zero from the requesting RTPM invites the accepting RTPM to select checkpoint-size. If this field is absent, checkpoint-size zero is assumed.

7.1.4.2 *Window-size*

The window-size field allows negotiation of the maximum number of outstanding minor synchronization points before data transfer shall be suspended. If this field is absent, window-size 3 is assumed.

7.1.4.3 *Dialogue-mode*

This is the dialogue-mode parameter value from the RT-OPEN request primitive. It appears as the dialogue-mode parameter value of the RT-OPEN indication primitive.

The value of this field is either monologue, or two-way-alternate. If this field is absent, monologue is assumed.

7.1.4.4 *User-data*

This is the user-data parameter value from the RT-OPEN request primitive. It appears as the user-data paremeter value of the RT-OPEN indication primitive.

The value of this field is transparent to the RTPM.

7.1.4.5 *Session-connection-identifier*

This field is used only in the association-recovery procedure.

7.1.4.6 *Application-protocol*

This field is used solely in the X.410-1984 mode. It is the application-protocol parameter value from the RT-OPEN request primitive. It appears as the application-protocol parameter value in the RT-OPEN indication primitive.

7.1.5 *Use of the RTOAC APDU fields*

The RTOAC APDU fields are used as follows.

7.1.5.1 *Checkpoint-size*

The checkpoint-size field allows negotiation of the maximum amount of data (in units of 1024 octets) that may be sent between two minor synchronization points. If the checkpoint-size in the RTORQ APDU is greater than zero, the accepting RTPM shall supply a value in the RTOAC APDU that is less than or equal to the value in the RTORQ APDU, else the accepting RPTM may select checkpoint-size. A value of zero from the accepting RTPM indicates that checkpointing will not be used. The value of this field becomes the agreed maximum value and governs both directions of transfer. If this field is absent, it is assumed that checkpointing will not be used.

7.1.5.2 *Window-size*

This field is only used if checkpoint-size of the RTOAC APDU is greater than zero. The window-size field allows negotiation of the maximum number of outstanding minor synchronization points before data transfer shall be suspended. The accepting RTPM shall supply a value that is less than or equal to the value in the RTORQ APDU. This becomes the agreed maximum size and governs both directions of transfer. If this field is absent, window-size 3 is assumed.

7.1.5.3 *User-data*

This is the user-data parameter value from the RT-OPEN response primitive. It appears as the user-data parameter value of the RT-OPEN confirm service primitive.

The value of this field is transparent to the RTPM.

7.1.5.4 *Session-connection-identifier*

This field is used only in the association-recovery procedure.

7.1.6 *Use of the RTORJ APDU fields*

The RTORJ APDU fields are used as follows.

7.1.6.1 *Refuse-reason*

The refuse-reason field is only used in the X.410-1984 mode.

This field may contain one of the following values:

– rts-busy
The accepting RTPM, or the acceptor, is so loaded that it cannot support a new application-association. The requesting RTPM should retry after a period of time. This value is either provided by the accepting RTPM, or is derived from the result parameter value "rejected (transient)" of the RT-OPEN response primitive from the acceptor. It appears as the result parameter value "rejected (transient)" of the RT-OPEN confirm primitive to the requestor.

– cannot recover
This value is used only by the accepting RTPM in the association-recovery procedure if it is unable to accept an association-recovery.

– validation failure
The acceptor does not recognize the requestor's credentials as being valid for the proposed application-association. This value is the user-data parameter value of the RT-OPEN response primitive from the acceptor. It appears as the user-data parameter value of the RT-OPEN confirm primitive to the requestor.

– unacceptable-dialogue-mode
The acceptor does not accept the type of dialogue mode proposed for the application-association. This value is the user-data parameter value of the RT-OPEN response primitive from the acceptor. It appears as the user-data parameter value of the RT-OPEN confirm primitive to the requestor.

7.1.6.2 *User-data*

This field is only used in the normal mode:

This is the user-data parameter value of the RT-OPEN response primitive from the acceptor. It appears as the user-data parameter value of the RT-OPEN confirm primitive to the requestor.

The value of this field is transparent to the RTPM.

7.2 *Association-release*

7.2.1 *Purpose*

The association-release procedure is used for the normal release of an application-association by the association-initiator without loss of information in transit.

7.2.2 *APDUs used*

No APDUs are used in this procedure.

7.2.3 *Association-release procedure*

This procedure is driven by the following events:

a) an RT-CLOSE request primitive from the requestor (association-initiator);

b) an A-RELEASE indication primitive;

c) an RT-CLOSE response primitive from the acceptor (association-responder);

d) an A-RELEASE confirm primitive.

7.2.3.1 *RT-CLOSE request primitive*

The requestor may issue an RT-CLOSE request primitive only if it possesses the turn and if there is no outstanding RT-TRANSFER confirm primitive. When an RT-CLOSE request primitive is received from the requestor, the requesting (association-initiating) RTPM issues an A-RELEASE request primitive. The reason parameter of the A-RELEASE request primitive is the reason parameter of the RT-CLOSE request primitive. The user information parameter of the A-RELEASE request primitive is the user-data parameter of the RT-CLOSE request primitive.

Note – No RT-CLOSE request primitive parameters are present in X.410-1984 mode.

The requesting RTPM waits for a primitive from the ACSE service-provider and does not accept any other primitive from the requestor.

7.2.3.2 *A-RELEASE indication primitive*

The accepting RTPM receives the A-RELEASE indication primitive.

It issues an RT-CLOSE indication primitive to the acceptor. The RT-CLOSE indication parameter values are derived from the A-RELEASE indication primitive.

Note – No RT-CLOSE indication primitive parameters are present in X.410-1984 mode.

The RTPM waits for a primitive from the acceptor or the used service provider.

7.2.3.3 *RT-CLOSE response primitive*

When the accepting RTPM receives an RT-CLOSE response primitive, the accepting RTPM issues an A-RELEASE response primitive. The reason parameter of the A-RELEASE response primitive is the reason parameter of the RT-CLOSE response primitive. The user information parameter of the A-RELEASE response primitive is the user-data parameter of the RT-CLOSE response primitive. The result parameter value of the A-RELEASE response primitive is "affirmative".

Note – No RT-CLOSE response primitive parameters are present in X.410-1984 mode.

7.2.3.4 *A-RELEASE confirm primitive*

The requesting RTPM receives an A-RELEASE confirm primitive.

The requesting RTPM issues an RT-OPEN confirm primitive to the acceptor. The RT-OPEN confirm primitive parameter values are derived from the A-RELEASE confirm primitive.

Note – No RT-CLOSE confirm primitive parameters are present in X.410-1984 mode.

7.3 *Transfer*

7.3.1 *Purpose*

The transfer procedure is used to transfer an RTSE-user APDU from the requestor (sender) to the acceptor (receiver).

7.3.2 *APDUs used*

Each RTSE-user APDU, conveyed in an RT-TRANSFER request, constitutes an activity. For each application-association, at most one activity, or one interrupted activity awaiting resumption, may exist at a time.

The RTSE-user APDU value is transformed into the encoded-APDU-value and vice versa by means of the local syntax-matching services. The transfer procedure uses the RT-TRANSFER (RTTR) APDU. The transfer procedure supports segmenting and reassembling of the encoded-APDU-value into/from one or more RTTR APDUs.

An encoded-APDU-value is transferred as a single RTTR APDU if checkpointing is not used. Otherwise, the encoded-APDU-value is transferred as a series of RTTR APDUs, the maximum size (i.e. number of octets forming the RTTR APDU value) of each being the negotiated checkpoint-size. The concatenation of the RTTR APDU values is the encoded-APDU-value.

The fields of the RTTR APDU are listed in Table 5/X.228.

TABLE 5/X.228

RTTR APDU Fields

Field name	Presence	Source	Sink
User-data-part	M	req	ind/conf

7.3.3 *Transfer procedure*

This procedure is driven by the following events:

a) an RT-TRANSFER request primitive from the requestor (sender);

b) a P-ACTIVITY-START indication primitive, followed by one or more RTTR APDUs as user-data of P-DATA indication primitives each, except the last, followed by a P-MINOR-SYNCHRONIZE indication primitive;

c) a P-MINOR-SYNCHRONIZE confirm primitive;

d) a P-ACTIVITY-END indication primitive;

e) a P-ACTIVITY-END confirm primitive;

f) a transfer time-out.

7.3.3.1 *RT-TRANSFER request primitive*

If the requesting RTPM possesses the turn and receives a RT-TRANSFER request from the requestor, the requesting RTPM transforms the RTSE-user APDU value into the encoded-APDU-value by means of the encoding service of the local syntax-matching services.

The requesting RTPM issues a P-ACTIVITY-START request primitive and may start transmitting the first RTTR APDU in a P-DATA request primitive immediately after the P-ACTIVITY-START request primitive is issued, since the latter service is not a confirmed service.

The maximum RTTR APDU size will have been negotiated during the association-establishment procedure. The requesting RTPM shall submit, in P-DATA request primitives, RTTR APDUs that conform to that agreement. Checkpoints may only be inserted if a checkpoint-size greater than zero was negotiated during the association-establishment procedure.

If a transferred RTTR APDU is not the last in a series of RTTR APDUs used to transfer a single encoded-APDU-value, the requesting RTPM inserts a checkpoint by issuing a P-MINOR-SYNCHRONIZE request primitive. The requesting RTPM uses only the "explicit confirmation expected" type of minor synchronization. The requesting RTPM may issue further P-DATA request primitives and P-MINOR-SYNCHRONIZE request primitives unless the agreed window-size has been reached.

If the RTTR APDU is the only one, or the last in a series of RTTR APDUs used to transfer a single encoded-APDU-value, the requesting RTPM issues a P-ACTIVITY-END request primitive.

Consecutive P-DATA request primitives shall not be issued, and all data transfer shall take place within an activity.

7.3.3.2 *P-ACTIVITY-START indication primitive, RTTR APDUs, and P-MINOR-SYNCHRONIZE indication primitives*

The accepting RTPM receives a P-ACTIVITY-START indication primitive, indicating the start of transfer of a RTSE-user APDU. The accepting RTPM receives an RTTR APDU as user-data of a P-DATA indication primitive.

If the RTTR APDU is not the last in a series of RTTR APDUs used to transfer a single encoded-APDU-value, the accepting RTPM receives a P-MINOR-SYNCHRONIZE indication primitive. If the accepting RTPM has secured the RTTR APDU, it issues a P-MINOR-SYNCHRONIZE response primitive.

7.3.3.3 *P-MINOR-SYNCHRONIZE confirmed primitive*

When the requesting RTPM receives a P-MINOR-SYNCHRONIZE confirm primitive, it assumes that the accepting RTPM has secured the encoded-APDU-value APDU up to that point.

The requesting RTPM may issue further P-DATA request primitives and P-MINOR-SYNCHRONIZE request primitives unless the agreed window-size has been reached. The window is advanced when a P-MINOR-SYNCHRONIZE confirm primitive is received by the requesting RTPM.

When a complete encoded-APDU-value has been transmitted, the requesting RTPM issues a P-ACTIVITY-END request primitive.

7.3.3.4 *P-ACTIVITY-END indication primitive*

A P-ACTIVITY-END indication primitive indicates to the accepting RTPM that a complete encoded-APDU-value has been transferred. The accepting RTPM transforms the encoded-APDU-value into the RTSE-user APDU value by means of the decoding service of the local syntax-matching-services.

If the accepting RTPM has secured the complete RTSE-user APDU, it issues an RT-TRANSFER indication primitive to the acceptor, and issues a P-ACTIVITY-END response primitive.

The accepting RTPM records the session-connection-identifier and the activity identifier of the last RTSE-user APDU which it completely secured for association-recovery purposes.

7.3.3.5 *P-ACTIVITY-END confirm primitive*

An activity end is an implicit major synchronization point and once successfully confirmed by means of an P-ACTIVITY-END confirm primitive, it indicates to the requesting RTPM that the RTSE-user APDU has been secured by the accepting RTPM. The requesting RTPM may then delete the transferred RTSE-user APDU.

When the requesting RTPM receives the P-ACTIVITY-END confirm primitive, it issues an RT-TRANSFER confirm primitive with a result parameter value of "APDU-transferred" to the requestor.

7.3.3.6 *Transfer time-out*

If an APDU has not been transferred within the time specified in the transfer-time parameter of the RT-TRANSFER request primitive (that is, the requesting RTPM has not received the P-ACTIVITY-END confirm primitive), the requesting RTPM performs the transfer-discard procedure followed by the transfer-abort procedure.

If during the transfer-discard procedure, the requesting RTPM does not receive a P-ACTIVITY-DISCARD confirm primitive within a (locally specified) reasonable time, the requesting RTPM performs the transfer-abort procedure followed by the provider-abort procedure.

7.4 *Turn-please*

7.4.1 *Purpose*

The turn-please procedure is used by a receiver (requestor) to request the turn from the sender (acceptor).

7.4.2 *APDUs used*

The turn-please procedure uses the RT-TURN-PLEASE (RTTP) APDU.

The fields of the RTTP APDU are listed in Table 6/X.228.

TABLE 6/X.228

RTTP APDU Fields

Field name	Presence	Source	Sink
Priority	U	req	ind

7.4.3 *Turn-please procedure*

This procedure is driven by the following events:

a) am RT-TURN-PLEASE request primitive from the requestor;

b) an RTTP APDU as user-data of a P-TOKEN-PLEASE indication primitive.

7.4.3.1 *RT-TURN-PLEASE request primitive*

If the requesting RTPM does not possess the turn and receives an RT-TURN-PLEASE request from the requestor, the requesting RTPM issues a P-TOKEN-PLEASE request primitive. If the priority parameter is present in the RT-TURN-PLEASE request primitive, and RTTP APDU is formed from the parameter value and transferred as user-data of the P-TOKEN-PLEASE request primitive. This procedure may be performed either inside or outside an activity.

7.4.3.2 *RTTP APDU*

If the accepting RTPM receives a P-TOKEN-PLEASE indication primitive, the accepting RTPM issues an RT-TURN-PLEASE indication primitive to the acceptor. If an RTTP APDU is transferred as user-data of the P-TOKEN-PLEASE indication primitive, the RT-TURN-PLEASE indication primitive parameter is present and derived from the RTTP APDU.

7.4.4 *Use of the RTTP fields*

The RTTP APDU fields are used as follows.

7.4.4.1 *Priority*

This is the priority parameter value of the RT-TURN-PLEASE request primitive. It appears as the priority parameter value of the RT-TURN-PLEASE indication primitive.

The value of this field is transparent to the RTPM.

7.5 *Turn-give*

7.5.1 *Purpose*

The turn-give procedure is used by a sender (requestor) to give the turn to the receiver (acceptor). The requestor becomes the receiver and the acceptor becomes the sender.

7.5.2 *APDUs used*

No APDUs are used in this procedure.

7.5.3 *Turn-give procedure*

The turn-give procedure is driven by the following events:

a) an RT-TURN-GIVE request primitive;

b) a P-CONTROL-GIVE indication primitive.

7.5.3.1 *RT-TURN-GIVE request primitive*

If the requesting RTPM possesses the turn and receives an RT-TURN-GIVE request primitive from the requestor, it issues a P-CONTROL-GIVE request primitive and becomes the receiving RTPM. This may be done only outside an activity.

7.5.3.2 *P-CONTROL-GIVE indication primitive*

If the accepting RTPM receives a P-CONTROL-GIVE indication primitive, the accepting RTPM issues an RT-TURN-GIVE indication primitive to the acceptor, and issues a P-CONTROL-GIVE response primitive. The accepting RTPM becomes the sending RTPM.

7.6 *Error reporting*

7.6.1 *User-exception-report*

7.6.1.1 *Purpose*

The user-exception-report procedure is used by the receiving RTPM to report an error situation to the sending RTPM.

7.6.1.2 *APDUs used*

No APDUs are used in this procedure.

7.6.1.3 *User-exception-report procedure*

This procedure is driven by the following events:

a) a receiving RTPM problem;

b) a P-U-EXCEPTION-REPORT indication primitive.

7.6.1.3.1 *Receiving RTPM problem*

If the receiving RTPM detects a problem, it issues a P-U-EXCEPTION-REPORT request primitive and starts a local recovery timer. Depending on the severity of the detected error, the value of the reason parameter of the P-U-EXCEPTION-REPORT request primitive is as follows:

a) In severe problem situations, the velue "receiving ability jeopardized" is used.

b) In exceptional circumstances, the receiving RTPM may have to delete a partially received RTSE-user APDU, even though some minor synchromization points have been confirmed. In this case, the value "unrecoverable procedure error" is used.

c) If the receiving RTPM is not willing to complete a transfer procedure, the value "non-specific error" is used.

d) If the sending RTPM resumes a transfer procedure already finished by the receiving RTPM (see clause 7.8.1.3.2), the value "sequence error" is used.

e) For all other less severe error situations, the value "local SS-user error" is used.

7.6.1.3.2 *P-U-EXCEPTION-REPORT indication primitive*

If the sending RTPM receives a P-U-EXCEPTION-REPORT indication primitive, it performs one of the following procedures depending on the reason parameter value of the P-U-EXCEPTION-REPORT indication primitive:

a) With a value "receiving ability jeopardized", the transfer-abort procedure followed by the provider-abort procedure are performed.

b) With a value "unrecoverable procedure error", the transfer-discard procedure followed by transfer-retry procedure are performed.

c) With a value "non-specific error", the transfer-discard procedure followed by the transfer-abort procedure are performed.

d) With a value "sequence error", the transfer-discard procedure is performed and the requesting RTPM issues an RT-TRANSFER confirm primitive with a result parameter value of "APDU-transferred" to the requestor and the transfer procedure is finished.

e) With a value "local SS-user error" and at least one confirmed checkpoint in the transfer procedure, the transfer-interrupt procedure followed by the transfer-resumption procedure are performed. If no checkpoint was confirmed in the transfer procedure, the transfer-discard procedure followed by the transfer-retry procedure are performed.

7.6.2 *Provider-exception-report*

7.6.2.1 *Purpose*

If the presentation service-provider detects an unexpected situation during an activity, not covered by other services, a P-P-EXCEPTION-REPORT indication primitive is issued to both RTPMs.

7.6.2.2 *APDUs used*

No APDUs are used in this procedure.

7.6.2.3 *Provider-exception-report procedure*

This procedure is driven by the following events:

a) a P-P-EXCEPTION-REPORT indication primitive.

7.6.2.3.1 *P-P-EXCEPTION-REPORT indication primitive*

The receiving RTPM receives a P-P-EXCEPTION-REPORT indication primitive.

If the sending RTPM receives a P-P-EXCEPTION-REPORT indication primitive, it may perform one of the following procedures:

a) if at least one checkpoint was confirmed in the transfer procedure, the transfer-interrupt procedure followed by the transfer-resumption procedure; or,

b) if no checkpoint was confirmed in the transfer procedure, the transfer-discard procedure followed by the transfer-retry procedure; or,

c) the transfer-abort procedure followed by the provider-abort procedure.

7.7 *Error handling*

7.7.1 *Transfer-interrupt*

7.7.1.1 *Purpose*

The transfer-interrupt procedure is used by the sending RTPM to handle a less severe (than those handled by the other error handling procedures) error situation during the transfer procedure, if at least one checkpoint was confirmed during the transfer procedure.

7.7.1.2 *APDUs used*

No APDUs are used in this procedure.

7.7.1.3 *Transfer-interrupt procedure*

This procedure is driven by the following events:

a) a sending RTPM problem;

b) a P-ACTIVITY-INTERRUPT indication primitive;

c) a P-ACTIVITY-INTERRUPT confirm primitive.

7.7.1.3.1 *Sending RTPM problem*

If the sending RTPM detects a less severe problem and at least one checkpoint was confirmed during the transfer procedure, it issues a P-ACTIVITY-INTERRUPT request primitive with one of the following reason parameter values:

a) "non-specific error", if the problem was indicated by an error reporting procedure;

b) "local SS-user error", if the problem is a local sending RTPM problem.

7.7.1.3.2 *P-ACTIVITY-INTERRUPT indication primitive*

If the receiving RTPM receives a P-ACTIVITY-INTERRUPT indication primitive, it issues a P-ACTIVITY-INTERRUPT response primitive and starts a local recovery timer.

7.7.1.3.3 *P-ACTIVITY-INTERRUPT confirm primitive*

If the sending RTPM receives a P-ACTIVITY-INTERRUPT confirm primitive, it starts the transfer-resumption procedure.

7.7.2 *Transfer-discard*

7.7.2.1 *Purpose*

The transfer-discard procedure is used by the sending RTPM to escape from a more severe (than those handled by the transfer-interrupt procedure) error situation, or a less severe error situation if no checkpoint was confirmed, during the transfer procedure.

7.7.2.2 *APDUs used*

No APDUs are used in this procedure.

7.7.2.3 *Transfer-discard procedure*

This procedure is driven by the following events:

a) a sending RTPM problem;

b) a P-ACTIVITY-DISCARD indication primitive;

c) a P-ACTIVITY-DISCARD confirm primitive.

7.7.2.3.1 *Sending RTPM problem*

If the sending RTPM detects a more severe problem, or a less severe problem if no checkpoint was confirmed during the transfer procedure, it issues a P-ACTIVITY-DISCARD request primitive with one of the following Reason parameter values:

a) "non-specific error", if the problem was indicated by an error reporting procedure;

b) "local SS-user error", or "unrecoverable procedural error", if the problem is a local sending RTPM problem.

7.7.2.3.2 *P-ACTIVITY-DISCARD indication primitive*

If the receiving RTPM receives a P-ACTIVITY-DISCARD indication primitive, it issues a P-ACTIVITY-DISCARD response primitive. The receiving RTPM deletes all knowledge and contents of the associated RTSE-user APDU so far received.

If the receiving RTPM has already issued an RT-TRANSFER indication primitive, it performs the association-abort procedure. The abort-reason field value of the RTAB APDU is "transfer-completed". In this case the sending RTPM ends the transfer procedure with a positive RT-TRANSFER confirm primitive and the association-recovery procedure is performed.

7.7.2.3.3 *P-ACTIVITY-DISCARD confirm primitive*

The receipt of a P-ACTIVITY-DISCARD confirm primitive by the sending RTPM signifies the completion of the transfer-discard procedure.

7.7.3 *Association-abort*

7.7.3.1 *Purpose*

The association-abort procedure is used by the RTPMs to handle the most severe error situations. This procedure can be performed between an RT-TRANSFER request primitive and its corresponding RT-TRANSFER confirm primitive.

7.7.3.2 *APDUs used*

The association-abort procedure uses the RT-ABORT (RTAB) APDU. The fields of the RTAB APDU are listed in Table 7/X.228.

Note – The RTAB APDU is also used by the provider-abort and the user-abort procedure.

TABLE 7/X.228

RTAB APDU Fields

Field name	Presence	Source	Sink
Abort-reason	T	sp	sp
Reflected-parameter	T	sp	sp
User-data	U	req	ind

7.7.3.3 *Association-abort procedure*

This procedure is driven by the following events:

a) an RTPM-abort;

b) an RTAB APDU.

7.7.3.3.1 *RTPM-abort*

Either the receiving or the sending RTPM transfers an RTAB APDU to its peer as user-data of an A-ABORT request primitive. If the RTPM is the association-initiating RTPM, it performs the association recovery procedure. If the RTPM is the association-responding RTPM, it awaits association-recovery. The receiving RTPM starts a local recovery timer.

After successful association recovery, the sending RTPM performs the transfer-resumption procedure.

7.7.3.3.2 *RTAB APDU*

Either the sending RTPM or the receiving RTPM may receive an RTAB APDU as user-data of an A-ABORT indication primitive. If the RTPM is the association-initiating RTPM, it performs the association-recovery procedure. If the RTPM is the association-responding RTPM, it awaits association recovery. The receiving RTPM starts a local recovery timer.

After successful association recovery, the sending RTPM performs the transfer-resumption procedure.

7.7.3.4 *Use of the RTAB APDU fields*

The RTAB APDU fields are used as follows:

7.7.3.4.1 *Abort-reason*

This field may contain one of the following values:

– local-system-problem	
– invalid-parameter	The invalid parameters are specified in the reflected-parameter field.
– unrecognized-activity	The sending RTPM shall perform the transfer-abort procedure optionally followed by the provider-abort procedure.
– temporary-problem	No attempt at association-recovery should be made for a period of time determined by a local rule.
– protocol-error	Of the RTPM.
– permanent-error	This value is used solely by the provider-abort procedure in normal mode.
– user-abort	This value is used solely by the user-abort procedure in normal mode.
– transfer-completed	The receiving RTPM could not discard an already completed transfer.

7.7.3.4.2 *Reflected-parameter*

The reflected-parameter field is a bit string that identifies which parameters are regarded as invalid parameters in the primitive received from the used service by the aborting RTMP before the association-abort. The order of the bits in the bit string is the same as the order of the parameters in the tables of service parameters in Recommendations X.217 and X.216 (i.e. bit 1 represents the first parameter, etc.).

7.7.3.4.3 *User-data*

This field is not used in the association-abort procedure.

7.7.4 *Association-provider-abort*

7.7.4.1 *Purpose*

The association-provider-abort procedure is used to handle an ACSE-provider, or a presentation service-provider abort.

7.7.4.2 *APDUs used*

No APDUs are used in this procedure.

7.7.4.3 *Association-provider-abort procedure*

This procedure is driven by the following event:

a) an AP-ABORT indication primitive.

7.7.4.3.1 *AP-ABORT indication primitive*

An association-provider-abort is indicated to both RTPMs by an AP-ABORT indication primitive, and may occur at any time.

After such an event, the association-initiating RTPM starts the association-recovery procedure. Both RTPMs start a local recovery timer.

If the association-provider-abort procedure was performed during the transfer procedure, the sending RTPM starts the transfer-resumption procedure after the association-recovery procedure is successfully completed. If the association-recovery procedure was not successfully completed, the sending RTPM performs the transfer-error procedure, and the provider-abort procedure.

7.8 *Error recovery*

7.8.1 *Transfer-resumption*

7.8.1.1 *Purpose*

The transfer-resumption procedure is used by the sending RTPM to recover from:

a) an error situation handled by the transfer-interrupt procedure; or,

b) an error situation handled by the association-abort procedure during a transfer procedure. In this case the transfer-resumption procedure is performed after an association-recovery procedure is successfully performed. If no checkpoint was confirmed in the interrupted transfer procedure, the transfer-discard procedure followed by the transfer-retry procedure are performed after transfer-resumption.

7.8.1.2 *APDUs used*

The transfer-resumption procedure uses the RTTR APDU (see clause 7.3.2).

7.8.1.3 *Transfer-resumption procedure*

This procedure is driven by the following events:

a) the resumption of an interrupted activity;

b) a P-ACTIVITY-RESUME indication primitive.

After these events, the transfer procedure is used to continue (see clause 7.3.3).

7.8.1.3.1 *Resumption of an interrupted activity*

The sending RTPM issues a P-ACTIVITY-RESUME request primitive with parameters that link the resumed Activity to the previously interrupted one.

After the sending RTPM has issued the P-ACTIVITY-RESUME request primitive and at least one checkpoint was confirmed in the interrupted transfer procedure, it continues the transfer procedure by issuing a P-DATA request primitive for the RTTR APDU following the last confirmed checkpoint. If no checkpoint was confirmed in the interrupted transfer procedure, the transfer-discard procedure followed by the transfer-retry procedure are performed.

7.8.1.3.2 *P-ACTIVITY-RESUME indication primitive*

If the receiving RTPM receives a P-ACTIVITY-RESUME indication primitive, it checks the Old Activity Identifier and the Old Session Connection Identifier parameters of the P-ACTIVITY-RESUME indication primitive with the corresponding information (session-connection-identifier, and Activity Identifier) recorded for the last completely secured transfer (see clause 7.3.3.4).

If the information coincides, the receiving RTPM either:

a) responds correctly to the sending RTPM according to the transfer procedure, but discards the data it receives, and does not issue an RT-TRANSFER indication primitive; or,

b) performs the user-exception-report procedure with a Reason parameter value of "sequence error".

If the information does not coincide, and the old Activity Identifier and the old Session Connection Identifier parameters match the corresponding information of the previously interrupted activity, the transfer-resumption procedure continues as for the transfer procedure with a P-DATA indication primitive for the RTTR APDU following the last confirmed checkpoint.

If the receiving RTPM cannot resume the activity, the receiving RTPM performs the user-exception-report procedure or the association-abort procedure.

7.8.2 *Transfer-retry*

7.8.2.1 *Purpose*

The transfer-retry procedure is used by the sending RTPM to recover from an error situation handled by the transfer-discard procedure.

The completion of this procedure is as for the transfer procedure.

7.8.2.2 *APDUs used*

The transfer-retry procedure uses the RTTR APDU (see clause 7.3.2).

7.8.2.3 *Transfer-retry procedure*

The sending RTPM performs the transfer procedure (see clause 7.3.3). A new Activity Identifier parameter value is used in the P-ACTIVITY-START request primitive.

7.8.3 *Association-recovery*

7.8.3.1 *Purpose*

The association-recovery procedure is used by the association-initiating RTPM to recover from an error situation handled by the association-abort procedure or the association-provider-abort procedure.

7.8.3.2 *APDUs used*

The association-recovery procedure uses the RT-OPEN-REQUEST (RTORQ) APDU, the RT-OPEN-ACCEPT (RTOAC) APDU, and the RT-OPEN-REJECT (RTORJ) APDU.

7.8.3.2.1 *RTORQ APDU*

The RT-OPEN-REQUEST (RTORQ) APDU is used in the request to recover an application-association. The fields of the RTORQ APDU are listed in clause 7.1.2.1.

The following rules apply:

a) the User-data field is not used;

b) the Session-connection-identifier field is mandatory.

7.8.3.2.2 *RTOAC APDU*

The RT-OPEN-ACCEPT (RTOAC) APDU is used in the positive response to the request to recover an application-association. The fields of the RTOAC APDU are listed in clause 7.1.2.2.

The following rules apply:

a) the User-data field is not used;

b) the Session-connection-identifier field is mandatory.

7.8.3.2.3 *RTORJ APDU*

The RT-OPEN-REJECT (RTORJ) APDU is used in the negative response to the request to recover an application-association. The fields of the RTORJ APDU are listed in clause 7.1.2.3.

The following rules apply:

a) the Refuse-reason field is used solely in the X.410-1984 mode;

b) the User-data field is not used.

7.8.3.3 *Association-recovery procedure*

This procedure is driven by the following events:

a) an A-ASSOCIATE request primitive by the association-initiating RTPM;

b) an RTORQ APDU as user-data on an A-ASSOCIATE indication primitive;

c) an A-ASSOCIATE confirm primitive that may contain either an RTOAC APDU, or an RTORJ, or no APDU.

7.8.3.3.1 *A-ASSOCIATE request primitive*

The association-initiating RTPM forms an RTORQ APDU from its internal data. The association-initiating RTPM issues an A-ASSOCIATE request primitive using information stored during the association-establishment procedure (see clause 7.1.3.1). The RTORQ APDU is the User Information parameter value of the A-ASSOCIATE request primitive.

The association-initiating RTPM waits for a primitive from the ACSE service-provider.

7.8.3.3.2 *RTORQ APDU*

If the application-association is not accepted by the ACSE service-provider, no A-ASSOCIATE indication primitive is received by the association-responding RTPM and no action takes place.

If the application-association is accepted by the ACSE service-provider, the association-responding RTPM receives the RTORQ APDU as the User Information parameter on an A-ASSOCIATE indication primitive.

If any of the A-ASSOCIATE indication primitive parameters, or any of the RTORQ APDU fields, is unacceptable to the association-responding RTPM, or if the association-responding RTPM is not able to accept the application-association, it forms and sends an RTORJ APDU with appropriate parameters from internal data. The association-responding RTPM issues an A-ASSOCIATE response primitive. The RTORJ APDU is sent as the User Information parameter of the A-ASSOCIATE response primitive. The application-association is not recovered.

If the A-ASSOCIATE indication primitive parameters, and the RTORQ APDU fields are acceptable to the association-responding RTPM, the association-responding RTPM forms an RTOAC APDU using internal data. The association-responding RTPM issues an A-ASSOCIATE response primitive. The RTOAC APDU is sent as the User Information parameter of the A-ASSOCIATE response primitive.

7.8.3.3.3 *A-ASSOCIATE confirm primitive*

The association-initiating RTPM receives an A-ASSOCIATE confirm primitive. The following situations are possible:

a) the association-recovery has been accepted;

b) the accepting RTPM has rejected the association-recovery; or

c) the ACSE service provider has rejected the association-recovery.

If the association-recovery was accepted, the A-ASSOCIATE confirm primitive Result parameter has the value "accepted" and the RTOAC APDU is the value of the User Information Parameter of the A-ASSOCIATE confirm primitive. The application-association is recovered successfully, and if the association-abort occurred during the transfer procedure, the sending RTPM continues with the transfer-resumption procedure.

If the association-recovery was rejected by the responding RTPM, the A-ASSOCIATE confirm primitive Result parameter has one of the values "rejected...", the A-ASSOCIATE confirm primitive Result Source parameter has the value "ACSE service-user" and the RTORJ APDU is the value of the User Information Parameter of the A-ASSOCIATE confirm primitive. The application-association is not recovered.

If the association-recovery was rejected by the ACSE service-provider, the A-ASSOCIATE confirm primitive Result parameter has one of the values "rejected..." and the A-ASSOCIATE confirm primitive Result Source parameter has either the value "ACSE service-provider" or "presentation service-provider". The application-association is not recovered.

If the application-association was not recovered, the association-recovery procedure is performed again by the association-initiating RTPM after a time determined by a local rule:

a) if the Result parameter of the A-ASSOCIATE confirm primitive has the following value "rejected (transient)"; or

b) if, in X.410-1984 mode, the Refuse-reason field of the RTORJ APDU has the value "rts-busy".

In all other cases a provider-abort is performed as follows.

If the association-initiating RTPM is the sending RTPM, and the association-abort occurred during the transfer procedure, the sending RTPM performs the transfer-abort procedure. The association-initiating RTPM performs the provider-abort procedure.

If the association-responding RTPM detects a recovery-time-out, the following actions take place. If the association-responding RTPM is the sending RTPM, and the association-abort occurred during the transfer procedure, the sending RTPM performs the transfer-abort procedure. The association-responding RTPM performs the provider-abort procedure.

7.8.3.4 *Use of the RTORQ APDU fields*

The RTORQ APDU fields are used as follows.

7.8.3.4.1 *Checkpoint-size*

See clause 7.1.4.1.

7.8.3.4.2 *Window-size*

See clause 7.1.4.2.

7.8.3.4.3 *Dialogue-mode*

See clause 7.1.4.3.

7.8.3.4.4 *User-data*

This field is not used in the association-recovery procedure.

7.8.3.4.5 *Session-connection-identifier*

The Session-connection-identifier is used to specify the original session connection used in the association-establishment procedure. This is used in order to relate the new session-connection to the existing application-association.

7.8.3.5 *Use of the RTOAC APDU fields*

The RTOAC APDU fields are used as follows.

7.8.3.5.1 *Checkpoint-size*

See clause 7.1.5.1.

7.8.3.5.2 *Window-size*

See clause 7.1.5.2.

7.8.3.5.3 *User-data*

This field is used only in the association-recovery procedure.

7.8.3.5.4 *Session-connection-identifier*

The Session-connection-identifier is used to specify the original session connection used in the association-establishment procedure. This is used in order to relate the new session-connection to the existing application-association.

8.3.6 *Use of the RTORJ APDU fields*

The RTORJ APDU fields are used as follows.

7.8.3.6.1 *Refuse-reason*

The Refuse-reason field is used solely in the X.410-1984 mode.

This field may contain one of the following values:

– rts-busy	The association-responding RTPM is so loaded that it cannot support the application-association. The association-initiating RTPM should retry after a period of time. This value is provided by the association-responding RTPM.
– cannot recover	This value is used by the association-responding RTPM, if it is unable to accept an association-recovery.

7.8.3.6.4 *User-data*

This field is not used in the association-recovery procedure.

7.9 *Abort*

These procedures are performed when a successful recovery from one of the error handling procedures is not possible.

7.9.1 *Transfer-abort*

7.9.1.1 *Purpose*

The transfer-abort procedure is used by the sending RTPM if the transfer of an RTSE-user APDU is not possible.

7.9.1.2 *APDUs used*

No APDUs are used in this procedure.

7.9.1.3 *Transfer-abort procedure*

The sending RTPM issues an RT-TRANSFER confirm primitive with a Result parameter value "APDU-not-transferred". The APDU parameter value is the RTSE-user APDU not transferred.

7.9.2 *Provider-abort*

7.9.2.1 *Purpose*

The provider-abort procedure is used by the RTPMs, if recovery is not possible.

7.9.2.2 *APDUs used*

If an application-association exists, the provider-abort procedure uses the RT-ABORT (RTAB) APDU. The RTAB APDU is specified in clause 7.7.3.2.

7.9.2.3 *Provider-abort procedure*

This procedure is driven by the following events:

a) an RTPM-abort;

b) an RTAB APDU;

c) local recovery time-out.

7.9.2.3.1 *RTPM-abort*

If an application-association exists, either the receiving or the sending RTPM transfers an RTAB APDU to its peer as the User-data parameter of an A-ABORT request primitive. The RTPM issues a RT-P-ABORT indication primitive to its RTSE-user.

7.9.2.3.2 *RTAB APDU*

If the sending or the receiving RTPM receives an RTAB APDU as the User-data parameter of an A-ABORT indication primitive, it issues an RT-P-ABORT indication primitive to its RTSE-user.

7.9.2.3.3 *Recovery time-out*

If an application-association does not exist and a local recovery time-out occurs, the RTPM issues an RT-P-ABORT indication primitive to its RTSE-user.

7.9.2.4 *Use of the RTAB APDU fields*

The RTAB APDU fields are used as follows.

7.9.2.4.1 *Abort-reason*

The value of this field is "permanent-error".

7.9.2.4.2 *Reflected-parameter*

This field is not used.

7.9.2.4.3 *User-data*

This field is not used.

7.9.3 *User-abort*

7.9.3.1 *Purpose*

The user-abort procedure is used by the requestor to abort an application-association.

7.9.3.2 *APDUs used*

The user-abort procedure uses the RT-ABORT (RTAB) APDU. The RTAB APDU is specified in clause 7.7.3.2.

7.9.3.3 *User-abort procedure*

This procedure is driven by the following events:

a) an RT-U-ABORT request primitive from the requestor;

b) an RTAB APDU as User-data of an A-ABORT indication primitive.

7.9.3.3.1 *RT-U-ABORT request*

If the requesting RTPM receives an RT-U-ABORT request primitive from the requestor, an RTAB APDU is formed from the parameter value of the RT-U-ABORT request primitive and transferred as User-data of an A-ABORT request primitive.

7.9.3.3.2 *RTAB APDU*

If the accepting RTPM receives the RTAB APDU as User-data of an A-ABORT indication primitive, the accepting RTPM issues an RT-U-ABORT indication primitive to the acceptor. The RT-U-ABORT primitive parameter is derived from the RTAB APDU.

7.9.3.4 *Use of the RTAB APDU fields*

The RTAB APDU fields are used as follows.

7.9.3.4.1 *Abort-reason*

The value of this field is “user-error”.

7.9.3.4.2 *Reflected-parameter*

This field is not used.

7.9.3.4.3 *User-data*

This is the User-data parameter value of the RT-U-ABORT request primitive. It appears as the User-data parameter value of the RT-U-ABORT indication primitive.

7.10 *Rules for extensibility*

In addition to the procedures stated above the following rule also applies when processing APDUs defined in this Recommendation:

– Ignore parameters which are not defined in this Recommendation for RTORQ, RTOAC, and RTORJ APDUs.

8 Mapping to used services

This clause defines how an RTPM transfers APDUs by means of:

a) the ACSE services, or

b) the presentation-service.

Clause 8.1 defines the mapping on the ACSE services, and clause 8.2 defines the mapping on the presentation-service.

Identification of the named abstract syntax in use is assumed for all RTSE services and is mapped onto used services, however this is a local matter and outside the scope of this Recommendation.

8.1 *Mapping on the ACSE services*

This clause defines how the ACSE service primitives described in Recommendation X.217 are used by the RTPM. Table 8/X.228 defines the mapping of the RTSE service primitives and APDUs to the ACSE service primitives.

TABLE 8/X.228

ACSE Mapping Overview

RTSE service	APDU	ACSE service
RT-OPEN request/indication	RTORQ	A-ASSOCIATE request/indication
RT-OPEN response/confirm	RTOAC	A-ASSOCIATE response/confirm
RT-OPEN response/confirm	RTORJ	A-ASSOCIATE response/confirm
RT-CLOSE request/indication	–	A-RELEASE request/indication
RT-CLOSE response/confirm	–	A-RELEASE response/confirm
association-abort	RTAB	A-ABORT request/indication
association-provider-abort	–	A-P-ABORT indication
RT-P-ABORT indication	RTAB	A-ABORT request/indication
RT-U-ABORT request/indication	RTAB	A-ABORT request/indication

Clause 8.1.1 defines the mapping onto ACSE in normal mode. Clause 8.1.2 defines the mapping onto ACSE in X.410-1984 mode.

8.1.1 *Mapping on the ACSE services in normal mode*

8.1.1.1 *Association-establishment procedure*

The association-establishment procedure takes place concurrently with the underlying ACSE association establishment.

8.1.1.1.1 *Directly mapped parameters*

The following parameters of the RT-OPEN service primitives are mapped directly onto the corresponding parameters of the A-ASSOCIATE service primitives:

a) Mode

b) Application Context Name

c) Calling AP Title

d) Calling AP Invocation-identifier

e) Calling AE Qualifier

f) Calling AE Invocation-identifier

g) Called AP Title

h) Called AP Invocation-identifier

i) Called AE Qualifier

j) Called AE Invocation-identifier

k) Responding AP Title

l) Responding AP Invocation-identifier

m) Responding AE Qualifier

n) Responding AE Invocation-identifier

o) Result Source

p) Diagnostic

q) Calling Presentation Address

r) Called Presentation Address

s) Responding Presentation Address

t) Presentation Context Definition List

u) Presentation Context Definition Result List

v) Default Presentation Context Name

w) Default Presentation Context Result.

8.1.1.1.2 *Parameters not used*

The following parameters of the A-ASSOCIATE service primitives are not used:

a) Presentation Requirements

b) Initial Synchronization Point Serial Number.

8.1.1.1.3 *Use of the other A-ASSOCIATE request and indication primitive parameters*

8.1.1.1.3.1 *User information*

For both the A-ASSOCIATE request and indication primitives, the User Information parameter is used to carry the RTORQ APDU.

8.1.1.1.3.2 *Quality of service*

The parameters “Extended Control” and “Optimized Dialogue Transfer” are set to ”not required.” The remaining parameters are set such that default values are used.

8.1.1.1.3.3 *Session requirements*

This parameter is set by the association-initiating RTPM to select the following functional units:

a) Half-duplex functional unit

b) Exceptions functional unit

c) Minor Synchronize functional unit

d) Activity Management functional unit.

8.1.1.1.3.4 *Initial assignment of tokens*

The association-initiating RTPM will always request that the data token be available for either monologue or two-way alternate interactions.

The association-initiating RTPM will specify which RTPM will initially hold the data token (minor synchronize token and major/activity token) upon successful completion of the session-connection phase, according to the initial-turn parameter of the RT-OPEN request primitive.

The association-initiating RTPM shall assign all of the tokens to the same RTPM. The application-association may be rejected if this rule is violated. At any particular point in time, the holder of the tokens is referred to as the sending RTPM, the other as the receiving RTPM.

8.1.1.1.3.5 *Session connection identifier*

The association-initiating RTPM will supply a Session Connection Identifier, which will be used to uniquely identify the session-connection. This identifier is formed of the following components: SS-User Reference, Common Reference, and, optionally, Additional Reference Information. The SS-User Reference is conveyed as the Calling SS-User Reference by the association-initiating RTPM. Common Reference and Additional Reference Information are conveyed in similarly named parameters of the P-CONNECT primitive.

Each component, when present, will contain a data element of the appropriate type from the following definitions:

```
CallingSSuserReference :: = CHOICE{     T61String
                                           -- solely in X.410-1984 mode --,
                                        OCTET STRING
                                           -- solely in normal mode -- }

CommonReference :: = UTCTime

AdditionalReferenceInformation :: = T61String
```

8.1.1.1.4 *Use of the other A-ASSOCIATE response and confirm primitive parameters*

8.1.1.1.4.1 *User information*

Note – This parameter only has relevance if the application-association is accepted by the ACSE service-provider.

For both the A-ASSOCIATE response and confirm primitives, the User Information parameter is used to carry the RTOAC APDU, if the application-association is accepted; or the RTORJ APDU, if the application-association is rejected by either the association-responding RTPM, or the association-responder.

8.1.1.1.4.2 *Result*

For the A-ASSOCIATE response primitive the Result parameter is set by the association-responding RTPM as follows:

a) If the association-responding RTPM rejects the application-association, the value of this parameter is set to either “rejected (transient)” or “rejected (permanent)”.

b) If the association-responding RTPM accepts the request, the value of this parameter is derived from the Result parameter of the RT-OPEN response primitive.

8.1.1.1.4.3 *Quality of service*

This parameter has the same value as in the A-ASSOCIATE request and indication primitives.

8.1.1.1.4.4 *Session requirements*

This parameter has the same value as in the A-ASSOCIATE request and indication primitives.

8.1.1.1.4.5 *Initial assignment of tokens*

This parameter is not used.

8.1.1.1.4.6 *Session connection identifier*

This parameter has the same value as in the A-ASSOCIATE indication primitive. The Calling SS-User Reference value of the A-ASSOCIATE indication primitive is returned as Called SS-User Reference by the association-responding RTPM.

8.1.1.2 *Association-release procedure*

The association-release procedure takes place concurrently with the underlying ACSE association release.

8.1.1.2.1 *Directly mapped parameters*

The following parameters of the RT-CLOSE service primitives are mapped directly onto the corresponding parameters of the A-RELEASE service primitives:

a) Reason

b) User-data (on User Information)

8.1.1.2.2 *Use of the other A-RELEASE response and confirm primitive parameters*

8.1.1.2.2.1 *Result*

The value of this parameter is “affirmative”.

8.1.1.3 *Association-provider-abort*

8.1.1.3.1 *Use of the A-P-ABORT indication primitive parameters*

The use of the A-P-ABORT indication primitive parameters are defined in Recommendation X.217.

8.1.1.4 *Association-recovery procedure*

The association-recovery procedure takes place concurrently with the underlying ACSE association establishment.

8.1.1.4.1 *Parameters from RT-OPEN service*

The following parameters of the RT-OPEN service primitives are stored by the RTPMs, and mapped directly onto the corresponding parameters of the A-ASSOCIATE service primitives:

a) Mode

b) Application Context Name

c) Calling AP Title

d) Calling AP Invocation-identifier

e) Calling AE Qualifier

f) Calling AE Invocation-identifier

g) Called AP Title
h) Called AP Invocation-identifier
i) Called AE Qualifier
j) Called AE Invocation-identifier
k) Responding AP Title
l) Responding AP Invocation-identifier
m) Responding AE Qualifier
n) Responding AE Invocation-identifier
o) Calling Presentation Address
p) Called Presentation Address
q) Responding Presentation Address
r) Presentation Context Definition List
s) Presentation Context Definition Result List
t) Default Presentation Context Name
u) Default Presentation Context Result.

8.1.1.4.2 *Parameters not used*

The following parameters of the A-ASSOCIATE service primitives are not used:

a) Presentation Requirements
b) Initial Synchronization Point Serial Number.

8.1.1.4.3 *Parameters used as in the association-establishment procedure*

The following parameters of the A-ASSOCIATE service primitives are used as described for the association-establishment procedure (see clause 8.1.1.1):

a) User Information
b) Quality of service
c) Session Requirements
d) Session Connection Identifier.

8.1.1.4.4 *Use of the other A-ASSOCIATE request and indication primitive parameters*

8.1.1.4.4.1 *Initial assignment of tokens*

The following rules apply:

a) If the association-initiating RTPM has the Turn, it specifies the value "requestor side".
b) If the association-initiating RTPM does not have the Turn, but has issued a P-CONTROL-GIVE request primitive with no confirmation that the tokens were received, it specifies the value "acceptor side". (Receipt of data serves as confirmation that the tokens were received.)
c) If the association-initiating RTPM does not have the tokens and does not have a P-CONTROL-GIVE request primitive outstanding, it specifies the value "acceptor chooses".

8.1.1.4.5 *Use of the other A-ASSOCIATE response and confirm primitive parameters*

8.1.1.4.5.1 *Initial assignment of tokens*

If the value of this parameter in the A-ASSOCIATE indication primitive was "acceptor chooses", the association-responding RTPM will either keep (value "acceptor side") or return (value "requestor side") the tokens depending upon whether it had them before the session-connection was aborted.

8.1.1.4.5.2 *Result*

If the association-responding RTPM rejects the application-association, the value of this parameter is set to either "rejected (transient)" or "rejected (permanent)", else it is set to "accepted".

8.1.1.5 *Association-abort, provider-abort and user-abort procedures*

8.1.1.5.1 *Use of the A-ABORT request and indication primitive parameters*

8.1.1.5.1.1 *Abort source*

This parameter value is "requestor".

8.1.1.5.1.2 *User information*

This parameter value is the RTAB APDU.

8.1.2 *Mapping on the ACSE services in X.410-1984 Mode*

8.1.2.1 *Association-establishment procedure*

The association-establishment procedure takes place concurrently with the underlying ACSE association establishment.

8.1.2.1.1 *Directly mapped parameters*

The following parameters of the RT-OPEN service primitives are mapped directly onto the corresponding parameters of the A-ASSOCIATE service primitives:

a) Mode

b) Result Source

c) Diagnostic

d) Calling Presentation Address

e) Called Presentation Address

f) Responding Presentation Address

8.1.2.1.2 *Parameters not used*

The following parameters of the A-ASSOCIATE service primitives are not used:

a) Application Context Name

b) Calling AP Title

c) Calling AP Invocation-identifier

'd) Calling AE Qualifier

e) Calling AE Invocation-identifier

f) Called AP Title

g) Called AP Invocation-identifier

h) Called AE Qualifier

i) Called AE Invocation-identifier

j) Responding AP Title

k) Responding AP Invocation-identifier

l) Responding AE Qualifier

m) Responding AE Invocation-identifier

n) Presentation Context Definition List

o) Presentation Context Definition Result List

p) Default Presentation Context Name

q) Default Presentation Context Result.

8.1.2.1.3 *Parameters used as in normal mode*

The following parameters of the A-ASSOCIATE service primitives are used as in the normal mode (see clause 8.1.1):

a) User Information

b) Result

c) Quality of Service

d) Session Requirements

e) Initial Assignment of Tokens

f) Session Connection Identifier.

8.1.2.2 *Association-release procedure*

The association-release procedure takes place concurrently with the underlying ACSE association release.

8.1.2.2.1 *Parameters not used*

The following parameters of the A-RELEASE service primitives are not used:

a) Reason

b) User Information

8.1.2.3 *Association-provider-abort procedure*

8.1.2.3.1 *Use of the A-P-ABORT indication primitive parameters*

The use of the A-P-ABORT indication primitive parameters are defined in Recommendation X.217.

8.1.2.4 *Association-recovery procedure*

The association-recovery procedure takes place concurrently with the underlying ACSE association establishment.

8.1.2.4.1 *Parameters from RT-OPEN service*

The following parameters of the RT-OPEN service primitives are stored by the RTPMs, and mapped directly onto the corresponding parameters of the A-ASSOCIATE service primitives:

a) Mode

b) Calling Presentation Address

c) Called Presentation Address

d) Responding Presentation Address.

8.1.2.4.2 *Parameters not used*

The following parameters of the A-ASSOCIATE service primitives are not used:

a) Application Context Name

b) Calling AP Title

c) Calling AP Invocation-identifier

d) Calling AE Qualifier
e) Calling AE Invocation-identifier
f) Called AP Title
g) Called AP Invocation-identifier
h) Called AE Qualifier
i) Called AE Invocation-identifier
j) Responding AP Title
k) Responding AP Invocation-identifier
l) Responding AE Qualifier
m) Responding AE Invocation-identifier
n) Presentation Context Definition List
o) Presentation Context Definition Result List
p) Default Presentation Context Name
q) Default Presentation Context Result
r) Presentation Requirements
s) Initial Synchronization Point Serial Number.

8.1.2.4.3 *Parameters used as in normal mode*

The following parameters of the A-ASSOCIATE service primitives are used as in the normal mode (see clause 8.1.1):

a) User Information
b) Result
c) Quality of service
d) Session Requirements
e) Initial Assignment of Tokens
f) Session Connection Identifier.

8.1.2.5 *Association-abort, provider-abort and user-abort procedures*

8.1.2.5.1 *Parameters not used*

The following parameters of the A-ABORT service primitives are not used:

a) Abort Source

8.1.2.5.2 *Parameters used as in normal mode*

The following parameters of the A-ASSOCIATE service primitives are used as in the normal mode (see clause 8.1.1):

a) User Information

8.2 *Mapping on the presentation services*

This clause defines how the presentation service primitives described in Recommendation X.216 are used by the RTPM. Table 9/X.228 defines the mapping of the RTSE service primitives and APDUs on the presentation service primitives.

This Clause defines the mapping onto presentation services in both normal mode and in X.410-1984 mode.

8.2.1 *Transfer procedure*

8.2.1.1 *Use of the P-ACTIVITY-START request and indication primitive parameters*

8.2.1.1.1 *Activity identifier*

The Activity Identifier identifies the activity by means of a serial number. The first activity started on the session-connection is assigned the number 1. Each successive activity for that direction of transfer is assigned the next number. Thus numbering is separate for each direction of transfer.

The property required of Activity Identifiers is that they should uniquely identify an activity during a reasonable time interval within a particular session-connection, so that duplicates can be detected in the face of error situations. These identifiers are allocated by numbering the activities during a session, starting with one for the first and incrementing for each successive activity, and to represent the number by a data element of type INTEGER, encoded according to Recommendation X.209. It is unnecessary for the receiving RTPM to make assumptions on the allocation method, only to be able to compare two identifiers for equality, octet by octet.

TABLE 9/X.228

Presentation Mapping Overview

RTSE service	APDU	Presentation-service
RT-TRANSFER req	–	P-ACTIVITY-START req/ind
	RTTR	P-DATA req/ind
	–	P-MINOR-SYNCHRONIZE req/ind/resp/conf
RT-TRANSFER ind/conf	–	P-ACTIVITY-END req/ind/resp/conf
RT-TURN-PLEASE req/ind	RTTP	P-TOKEN-PLEASE req/ind
RT-TURN-GIVE req/ind	–	P-CONTROL-GIVE req/ind
user-exception-report	–	P-U-EXCEPTION-REPORT req/ind
provider-exception-report	–	P-P-EXCEPTION-REPORT ind
transfer-interrupt	–	P-ACTIVITY-INTERRUPT req/ind/resp/conf
transfer-discard	–	P-ACTIVITY-DISCARD req/ind/resp/conf
transfer-resumption	–	P-ACTIVITY-RESUME req/ind

req	request
ind	indication
resp	response
conf	confirm

8.2.1.1.2 *User data*

This parameter is not used.

8.2.1.2 *Use of the P-DATA request and indication primitive parameters*

8.2.1.2.1 *User data*

The maximum User data size (number of octets of the RTTR APDU value) will have been negotiated during the association-establishment procedure. The sending RTPM shall submit User data that conforms to that agreement.

8.2.1.3 *Use of the P-MINOR-SYNCHRONIZE service parameters*

8.2.1.3.1 *Type*

The RTPM uses only the "explicit confirmation expected" type of minor synchronization.

8.2.1.3.2 *Synchronization point serial number*

The session service-provider allocates checkpoint serial numbers and passes them to the sending and receiving RTPMs to associate with the transmitted data.

8.2.1.3.3 *User data*

This parameter is not used.

8.2.1.4 *Use of the P-ACTIVITY-END service parameters*

8.2.1.4.1 *Synchronization point serial number*

The serial number of the implied major synchronization point is allocated by the session service-provider and passed up to both RTPMs.

8.2.1.4.2 *User data*

This parameter is not used.

8.2.2 *Turn-please procedure*

8.2.2.1 *Use of the P-TOKEN-PLEASE request and indication primitive parameters*

8.2.2.1.1 *Tokens*

The receiving RTPM will only request the data token. Since the tokens cannot be separated, the sending RTPM always surrenders all of the other available tokens when issuing the P-CONTROL-GIVE request primitive.

8.2.2.1.2 *User data*

This is the RTTP APDU.

8.2.3 *Turn-give procedure*

8.2.3.1 *Use of the P-CONTROL-GIVE service parameters*

The P-CONTROL-GIVE service primitives have no parameters. The data, minor synchronize, and major/activity tokens are automatically passed to the other RTPM.

8.2.4 *User-exception-report procedure*

8.2.4.1 *Use of the P-U-EXCEPTION-REPORT service parameters*

8.2.4.1.1 *Reason*

This parameter may specify one of the following reasons:

a) receiving ability jeopardized

b) local SS-User error

c) sequence error

d) unrecoverable procedure error

e) non-specific error.

8.2.4.1.2 *User data*

This parameter is not used.

8.2.5 *Provider-exception-report procedure*

8.2.5.1 *Use of the P-P-EXCEPTION-REPORT service parameters*

8.2.5.1.1 *Reason*

One of the following reason codes shall be supplied:

a) protocol error

b) non-specific error.

8.2.6 *Transfer-interrupt Procedure*

8.2.6.1 *Use of the P-ACTIVITY-INTERRUPT service parameters*

8.2.6.1.1 *Reason*

This parameter may specify one of the following:

a) local SS-User error

b) non-specific error.

8.2.7 *Transfer-discard procedure*

8.2.7.1 *Use of the P-ACTIVITY-DISCARD service parameters*

8.2.7.1.1 *Reason*

This parameter may specify one of the following:

a) local SS-User error

b) unrecoverable procedure error

c) non-specific error.

8.2.8 *Transfer-resumption procedure*

8.2.8.1 *Use of the P-ACTIVITY-RESUME service parameters*

8.2.8.1.1 *Activity identifier*

The sending RTPM shall allocate and supply the next Activity Identifier number for the current session.

8.2.8.1.2 *Old activity identifier*

The sending RTPM shall supply the original Activity Identifier which was assigned to the previously interrupted activity in the P-ACTIVITY-START request primitive.

8.2.8.1.3 *Synchronization point serial number*

The sending RTPM will specify the Serial Number of the last confirmed checkpoint in the interrupted activity. The session service-provider will also set the current session serial number to this value. If there was no previously confirmed checkpoint, the activity cannot be continued. The sending RTPM shall then send a P-ACTIVITY-RESUME request primitive (with the Synchronization Point Serial Number set to zero), followed by a P-ACTIVITY-DISCARD request primitive.

8.2.8.1.4 *Old session connection identifier*

The sending RTPM may supply the Session Connection Identifier of theSession Connection during which the Activity was started; it shall supply it if that session connection is not the current one. This Session Connection Identifier is conveyed in the Calling SS-User Reference, Common Reference, and, optionally, Additional Reference Information components of this parameter. The Called SS-User Reference component is not used.

8.2.8.1.5 *User data*

This parameter is not used.

9 Abstract Syntax Definition of APDUs

The abstract syntax of each RTSE APDU is specified in this clause using the abstract syntax notation of Recommendation X.208, and is shown in Figure 1/X.228.

```
Reliable-Transfer-APDU {joint-iso-ccitt reliable-transfer (3) apdus (0)} DEFINITIONS ::=
BEGIN
EXPORTS rTSE, rTSE-abstract-syntax,
   RTORQapdu, RTOACapdu, RTORJapdu, RTABapdu;-- for use by Presentation Layer only
IMPORTS APPLICATION-SERVICE-ELEMENT FROM     Remote-Operations-Notation-extension
                                             {joint-iso-ccitt remote-operations(4)
                                             notation-extension(2)};
   rTSE-abstract-syntax OBJECT IDENTIFIER  ::={joint-iso-ccitt reliable-transfer(3) abstract-syntax(2)}
   rTSE APPLICATION-SERVICE-ELEMENT        ::={joint-iso-ccitt reliable-transfer(3) aseID (1)}
   RTSE-apdus ::=     CHOICE{
                           rtorq-apdu     [16] IMPLICIT RTORQapdu,
                           rtoac-apdu     [17] IMPLICIT RTOACapdu,
                           rtorj-apdu     [18] IMPLICIT RTORJapdu,
                           rttp-apdu                    RTTPapdu,
                           rttr-apdu                    RTTRapdu,
                           rtab-apdu      [22] IMPLICIT RTABapdu}
   -- Tags [19], [20], [21] are used by the values of the UNBIND macro of the RO-notation of
   -- Recommendation X.219. Tags [0] to [15] inclusive are reserved for the
   -- use by the APDUs of ROSE (Recommendation X.229). Any occurrence of
   -- ANY in this module shall be replaced by a single ASN.1 type (if any) in an RTSE-user
   -- protocol specification. In addition any RTSE-user protocol sharing a single named
   -- abstract syntax with the RTSE protocol shall use distinct tags for the single
   -- presentation data values in the user data parameters of the RT-CLOSE (if any) and
   -- RT-TRANSFER services. These tags shall be distinct from the tag values [16], [17],
   -- [18] and [22] and from the ASN.1 types INTEGER and OCTET STRING.
   -- Note - The above conditions are ensured, if the RTSE-user protocol specification uses the
   -- RO-notation of Recommendation X.229.
   -- In X.410-1984 mode only the components of RTORQapdu, RTOACapdu, RTORJapdu
   -- and RTABapdu are used by the presentation layer. This has the effect that the following
   -- APDU types appear in the protocol in X.410-1984 mode instead of the alternative types
   -- of the RTSE-apdus type:
   --                      RTORQapdu
   --                      RTOACapdu
   --                      RTORJapdu
   --                      RTTPapdu
   --                      RTTRapdu
   --                      RTABapdu
-- RTSE Protocol continued
```

FIGURE 1/X.228 (Part 1 of 3)

Abstract Syntax Specification of RTSE Protocol

```
– – RTSE Protocol continued

RTORQapdu ::=              SET{
   checkpointSize          [0] IMPLICIT INTEGER DEFAULT 0,
   windowSize              [1] IMPLICIT INTEGER DEFAULT 3,
   dialogueMode            [2] IMPLICIT INTEGER {monologue(0), twa(1)} DEFAULT monologue,
   connectionDataRQ        [3] ConnectionData,
   applicationProtocol     [4] IMPLICIT INTEGER OPTIONAL– –solely in X.410-1984 mode– –}

RTOACapdu ::=              SET{
   checkpointSize          [0] IMPLICIT INTEGER DEFAULT 0,
   windowSize              [1] IMPLICIT INTEGER DEFAULT 3,
   connectionDataAC        [2] ConnectionData}

RTORJapdu ::=              SET{
   refuseReason            [0] IMPLICIT RefuseReason OPTIONAL,– –only in X.410-1984 mode
   userDataRJ              [1] ANY OPTIONAL– –RTSE user data, only in normal mode– –}

RTTPapdu ::= – –priority– –    INTEGER

RTTRapdu ::=               OCTET STRING

RTABapdu ::=               SET{
   abortReason             [0] IMPLICIT AbortReason OPTIONAL,
   reflectedParameter      [1] IMPLICIT BIT STRING OPTIONAL,
                           – –8 bits maximum, only if abortReason is invalidParameter
   userdataAB              [2] ANY OPTIONAL  – –only in normal mode and if abortReason
                                             – –is userError– –}

– – RTSE Protocole continued
```

FIGURE 1/X.228 (Part 2 of 3)

Abstract Syntax Specification of RTSE Protocol

```
-- RTSE Protocol continued

ConnectionData ::=         CHOICE{
   open                      [0] ANY, -- RTSE user data
                                      -- this alternative is encoded as [0] IMPLICIT NULL
                                      -- in the case of absence of RTSE user data,

   recover                   [1] IMPLICIT SessionConnectionidentifier}

SessionConnectionIdentifier ::= SEQUENCE{
                             CallingSSuserReference,
                             CommonReference,
                             [0] IMPLICIT AdditionalReferenceInformation OPTIONAL}

RefuseReason ::=             INTEGER{

                                 rtsBusy(0),
                                 cannotRecover(1),
                                 validationFailure(2),
                                 unacceptableDialogueMode(3)}

CallingSSuserReference ::=   CHOICE{ T61String     -- solely in X.410-1984 mode --,
                                     OCTET STRING  -- solely in normal mode --}

CommonReference ::=              UTCTime

AdditionalReferenceInformation ::=  T61String

AbortReason ::=            INTEGER{
                           localSystemProblem(0),
                           invalidParameter(1), -- reflectedParameter supplied
                           unrecognizedActivity(2),
                           temporaryProblem(3),
                           -- the RTSE cannot accept a session for a period of time--
                           protocolError(4), -- RTSE level protocol error--
                           permanentProblem(5), -- provider-abort solely in normal mode--
                           userError(6), -- user-abort solely in normal mode--
                           transferCompleted(7), -- activity can't be discarded--}

END    -- of RTSE Protocol
```

FIGURE 1/X.228 (Part 3 of 3)

Abstract Syntax Specification of RTSE Protocol

10 Conformance

An implementation claiming conformance to this Recommendation shall comply with the requirements in clauses 10.1 through 10.3..

10.1 *Statement requirements*

An implementor shall state the following:

a) the application context for which conformance is claimed, including whether the system supports normal mode, X.410-1984 mode, or both.

10.2 *Static requirements*

The system shall:

a) conform to the abstract syntax definition of APDUs defined in clause 9.

10.3 *Dynamic requirements*

The system shall:

a) conform to the elements of procedure defined in clause 7,

b) conform to the mappings to the used services, for which conformance is claimed, as defined in clause 8.

ANNEX A

(to Recommendation X.228)

RTPM State Tables

This annex forms an integral Part of this Recommendation.

A.1 *General*

This annex defines a single Reliable Transfer Protocol Machine (RTPM) in terms of a state table. The state table shows the interrelationship between the state of an application-association, the incoming events that occur in the protocol, the actions taken, and, finally the resultant state of the application-association.

The RTPM state table does not constitute a formal definition of the RTPM. It is included to provide a more precise specification of the elements of procedure defined in clause 7.

This annex contains the following tables:

a) Table A-1/X.228 specifies the abbreviated name, source, and name/description of each incoming event. The sources are:
   1) RTSE-user (RTSE-user);
   2) peer RTPM (RTPM-peer);
   3) Association Control Service Element (ACSE);
   4) Presentation service-provider (PS-provider); and
   5) RTPM (RTPM).

b) Table A-2/X.228 specifies the abbreviated name of each state of the RTPM.

c) Table A-3/X.228 specifies the abbreviated name, target, and name/description of each outgoing event. The targets are:
   1) RTSE-user (RTSE-user);
   2) peer RTPM (RTPM-peer);
   3) Association Control Service Element (ACSE);
   4) Presentation service-provider (PS-provider); and
   5) RTPM (RTPM).

d) Table A-4/X.228 specifies the predicates.

e) Table A-5/X.228 specifies the specific actions.

f) Tables A-6/X.228 through A-16/X.228 including specifies the RTPM state table using the abbreviations of the above tables.

For some events the source and the target is the RTPM (internal event). If the RTPM issues an internal event as part of an action taken, the RTPM awaits that internal event in the resultant state.

A.2 *Conventions*

The intersection of an incoming event (row) and a state (column) forms a cell.

In the state table, a blank cell represents the combination of an incoming event and a state that is not defined for the RTPM (see § A.3.1). Some states await solely some incoming events from the source RTPM (internal events). These states are marked by * and no other incoming events are considered.

A non-blank cell represents an incoming event and a state that is defined for the RTPM. Such a cell contains one or more action lists. An action list may be either mandatory or conditional. If a cell contains a mandatory action list, it is the only action list in the cell.

A mandatory action list contains:

a) optionally one or more outgoing events,

b) optionally one or more specific actions, and

c) a resultant state.

A conditional action list contains:

a) a predicate expression comprising predicates and Boolean operators (¬ represents the Boolean NOT, & represents the Boolean AND), and

b) a mandatory action list (this mandatory list is used only if the predicate expression is true).

A local collision between an incoming event from the RTSE-user and the association-recovery procedure is modeled by deferring that event until completion of the association-recovery procedure.

A.3 *Actions to be taken by the RTPM*

The RTPM state table defines the action to be taken by the RTPM in terms of an optional outgoing event, optional specific actions, and the resultant state of the application-association.

A.3.1 *Invalid intersections*

Blank cells indicate an invalid intersection of an incoming event and state. If such an intersection occurs, one of the following actions is taken:

a) If the incoming event comes from the RTSE-user, or is an internal event, any action taken by the RTPM is a local matter.

b) If the incoming event is related to a received APDU, PS-provider, or ACSE; either the RTPM-issues an appropriate internal event, or the RTPM issues both an RT-PAind outgoing event (to its RTSE-user) and an RTAB outgoing event (to its peer RTPM).

A.3.2 *Valid intersections*

If the intersection of the state and incoming event is valid, one of the following actions is taken:

a) If the cell contains a mandatory action list, the RTPM takes the actions specified.

b) If a cell contains one or more conditional action lists, for each predicate expression that is true, the RTPM takes the actions specified. If none of the predicate expressions are true, the RTPM takes one of the actions defined in § A.3.1.

A.4 *Definition of variables and timers*

The following variables and timers are specified.

A.4.1 *Association-initiating RTPM*

This Boolean variable is set TRUE if the RTPM is the association-initiating RTPM (specific action [a1], else set FALSE (specific action [a2]).

This Boolean variable is tested in the predicate p11.

A.4.2 *Checkpoint-confirmed*

This Boolean variable is TRUE, if at least one checkpoint was confirmed during the transfer procedure. It is set FALSE at the beginning of the transfer procedure (specific actin [a30]). It is set TRUE, if a P-MINOR-SYNCHRONIZE confirm primitive is issued to the sending RTPM (specific action [a32]).

A.4.3 *Outstanding-minor-syncs*

This Integer variable indicates the number of outstanding checkpoint confirmations during the transfer procedure. It is set to zero at the beginning of the transfer procedure (specific action [a30] and [a33]). It is incremented by one, if a P-MINOR-SYNCHRONIZE request primitive is issued by the sending RTPM (specific action [a31]). It is decremented by 1, if a P-MINOR-SYNCHRONIZE confirm primitive is issued to the sending RTPM (specific action [a32]).

The value of this variable is compared with the value of the window-size field of the RTOAC APDU in the predicate p32. The value of this variable is compared with the value zero in predicate p33.

A.4.4 *Transfer timer Tr*

This timer is used to control the transfer time. It is set to the value of the transfer-time parameter of the RT-TRANSFER request primitive (specific action [a30]). It is reset if a RT-TRANSFER response primitive is issued by the sending RTPM (specific action [a35]).

In the case of time out the internal event tr-timeout occurs.

A.4.5 *Recovery timer rec*

This timer is used to control the recovery time. It is set to a locally specified value in the recovery case (specific action [a38]). It is reset after successful recovery (specific action [a39]).

In the case of time out the internal event rec-timeout occurs.

TABLE A-1/X.228 (Part 1 of 3)

Incoming Event List

Abbreviated name	Source	Name and description
RT-OPreq	RTSE-user	RT-OPEN request primitive
RT-OPres+	RTSE-user	RT-OPEN response primitive (Result = "accepted")
RT-OPres−	RTSE-user	RT-OPEN response primitive (Result = "rejected")
RT-CLreq	RTSE-user	RT-CLOSE request primitive
RT-CLres	RTSE-user	RT-CLOSE response primitive
RT-TRreq	RTSE-user	RT-TRANSFER request primitive
RT-TPreq	RTSE-user	RT-TURN-PLEASE request primitive
RT-TGreq	RTSE-user	RT-TURN-GIVE request primitive
RT-UAreq	RTSE-user	RT-U-ABORT request primitive
RTORQ	RTPM-peer	RTORQ APDU as user data of an A-ASSOCIATE indication primitive
RTOAC	RTPM-peer	RTOAC APDU as user data of an A-ASSOCIATE confirm primitive
RTORJ	RTPM-peer	RTORJ APDU as user data of an A-ASSOCIATE confirm primitive
RTAB	RTPM-peer	RTAB APDU as user data of an A-ABORT indication primitive
RTTR	RTPM-peer	RTTR APDU as user data of a P-DATA indication primitive
RTTP	RTPM-peer	P-TOKEN-PLEASE indication primitive optionally with RTTP APDU as user data

TABLE A-1/X.228 (Part 2 of 3)

Incoming Event List

Abbreviated name	Source	Name and description
A-ASCcnf–	ACSE	A-ASSOCIATE confirm primitive (Result = "rejected") no RTORJ APDU
A-RELind	ACSE	A-RELEASE indication primitive
A-RELcnf	ACSE	A-RELEASE confirm primitive
A-PABind	ACSE	A-P-ABORT indication primitive
P-ASind	PS-provider	P-ACTIVITY-START indication primitive
P-MSind	PS-provider	P-MINOR-SYNCHRONIZE indication primitive
P-MScnf	PS-provider	P-MINOR-SYNCHRONIZE confirm primitive
P-AEind	PS-provider	P-ACTIVITY-END indication primitive
P-AEcnf	PS-provider	P-ACTIVITY-END confirm primitive
P-CGind	PS-provider	P-CONTROL-GIVE indication primitive
P-UEind	PS-provider	P-U-EXCEPTION-REPORT indication primitive
P-PEind	PS-provider	P-P-EXCEPTION-REPORT indication primitive
P-AIind	PS-provider	P-ACTIVITY-INTERRUPT indication primitive
P-AIcnf	PS-provider	P-ACTIVITY-INTERRUPT confirm primitive
P-ADind	PS-provider	P-ACTIVITY-DISCARD indication primitive
P-ADcnf	PS-provider	P-ACTIVITY-DISCARD confirm primitive
P-ARind	PS-provider	P-ACTIVITY-RESUME indication primitive

TABLE A-1/X.228 (Part 3 of 3)

Incoming Event List

Abbreviated name	Source	Name and description
a-ab	RTPM	associated aborted, recover
a-res	RTPM	activity resumption by the receiving RTPM
a-ret	RTPM	activity completed, discarded, or interrupted
ass-ab	RTPM	start of association-abort procedure
ass-rec	RTPM	start of association-recovery procedure
ass-rec-neg	RTPM	association-recovery unsuccessful
next	RTPM	transfer of RTTR APDU
p-ab	RTPM	start of provider-abort procedure
r-problem-1	RTPM	receiving RTPM problem
r-problem-2	RTPM	receiving RTPM problem more severe than r-problem-1
rec-timeout	RTPM	recovery time out
rt-ab	RTPM	RTAB received
s-problem-1	RTPM	sending RTPM problem
s-problem-2	RTPM	sending RTPM problem more severe than s-problem-1
s-problem-3	RTPM	sending RTPM problem more severe than s-problem-2
tr-discard	RTPM	start of transfer-discard procedure
tr-interr	RTPM	start of transfer-interrupt procedure
tr-p-ab	RTPM	start of procedures transfer-abort followed by provider-abort
tr-pos	RTPM	transfer successful completed
tr-res	RTPM	start of transfer-resumption de transfert
tr-timeout	RTPM	transfer time out
transfer	RTPM	start of transfer or transfer-retry procedures
u-exr	RTPM	start of user-exception-report procedure

TABLE A-2/X.228 (Part 1 of 2)

RTPM States

Abbreviated name	Name and description
STA0	idle, unassociated
STA01	awaiting RTOAC, RTORJ, or A-ASCcnf−
STA02	awaiting RT-OPres+, or RT-OPres−
STA11	associated; RTPM is association-initiating RTPM and sending RTPM
STA12	associated; RTPM is association-initiating RTPM and receiving RTPM
STA21	associated; RTPM is association-responding RTPM and sending RTPM
STA22	associated; RTPM is association-responding RTPM and receiving RTPM
STA30	transfer; sending RTPM
STA31	suspended transfer; sending RTPM
STA32	awaiting P-AEcnf; sending RTPM
STA321*	awaiting tr-pos; sending RTPM
STA34*	awaiting tr-discard to be followed by RT-TRcnf+; sending RTPM
STA341	awaiting P-ADcnf to be followed by RT-TRcnf+; sending RTPM
STA35*	awaiting tr-discard to be followed by RT-TRcnf−; sending RTPM
STA351	awaiting P-ADcnf to be followed by RT-TRcnf−; sending RTPM
STA36*	awaiting tr-discard to be followed by transfer-retry procedure; sending RTPM
STA361	awaiting P-ADcnf to be followed by transfer-retry procedure; sending RTPM
STA37*	awaiting tr-interr to be followed by transfer-retry procedure; sending RTPM
STA371	awaiting P-AIcnf; sending RTPM
STA372*	awaiting tr-res; sending RTPM
STA38*	awaiting ass-ab; sending RTPM
STA381*	awaiting a-ab; transfer sending RTPM
STA39*	awaiting rt-ab; transfer sending RTPM
STA40	awaiting RTTR; transfer receiving RTPM
STA400	awaiting RTTR; ignored transfer receiving RTPM

TABLE A-2/X.228 (Part 2 of 2)

RTPM States

Abbreviated name	Name and description
STA41	awaiting P-MSind or P-AEind; transfer receiving RTPM
STA410	awaiting P-MSind or P-AEind; ignored transfer receiving RTPM
STA42	awaiting recovery after u-exr event; transfer receiving RTPM
STA43*	awaiting a-ret; transfer receiving RTPM
STA44*	awaiting u-exr; transfer receiving RTPM
STA45*	awaiting a-res; transfer receiving RTPM
STA48*	awaiting ass-ab; transfer receiving RTPM
STA481*	awaiting a-ab; transfer receiving RTPM
STA49*	awaiting rt-ab; transfer receiving RTPM
STA51*	awaiting ass-rec or ass-rec-neg; association-recovery procedure outside activity
STA510	awaiting RTOAC or RTORJ; association-recovery procedure outside activity
STA52	awaiting RTORQ; association-recovery procedure outside acitivity
STA53*	awaiting ass-rec or ass-rec-neg; association-recovery procedure sending RTPM
STA531	awaiting RTOAC or RTORJ; association-recovery procedure sending RTPM
STA532	awaiting RTORQ; association-recovery procedure sending RTPM
STA54*	awaiting ass-rec or ass-rec-neg; association-recovery procedure receiving RTPM
STA541	awaiting RTOAC or RTORJ; association-recovery procedure receiving RTPM
STA542	awaiting RTORQ; association-recovery procedure receiving RTPM
STA70*	awaiting abort; unassociated
STA71*	awaiting abort; associated
STA72*	awaiting rt-ab outside transfer
STA91	awaiting RT-CLres
STA92	awaiting A-RELcnf

TABLE A-3/X.228 (Part 1 of 3)

Outgoing Event List

Abbreviated name	Source	Name and description
RT-OPind	RTSE-user	RT-OPEN indication primitive
RT-OPcnf+	RTSE-user	RT-OPEN confirm primitive (Result = "accepted")
RT-OPcnf−	RTSE-user	RT-OPEN confirm primitive (Result = "rejected")
RT-CLind	RTSE-user	RT-CLOSE indication primitive
RT-CLcnf	RTSE-user	RT-CLOSE confirm primitive
RT-TRind	RTSE-user	RT-TRANSFER indication primitive
RT-TPind	RTSE-user	RT-TURN-PLEASE indication primitive
RT-TRcnf+	RTSE-user	RT-TRANSFER confirm primitive (Result = "APDU-transferred")
RT-TRcnf−	RTSE-user	RT-TRANSFER confirm positive (Result = "APDU-not-transferred")
RT-TGind	RTSE-user	RT-TURN-GIVE indication primitive
RT-UAind	RTSE-user	RT-U-ABORT indication primitive
RT-PAind	RTSE-user	RT-P-ABORT indication primitive
RTORQ	RTPM-peer	RTORQ APDU as user data of an A-ASSOCIATE request primitive
RTOAC	RTPM-peer	RTORQ APDU as user data of an A-ASSOCIATE response primitive
RTORJ	RTPM-peer	RTORJ APDU as user data of an A-ASSOCIATE response primitive
RTAB	RTPM-peer	RTAB APDU as user data of an A-ABORT request primitive
RTTR	RTPM-peer	RTTR APDU as user data of a P-DATA request primitive
RTTP	RTPM-peer	P-TOKEN-PLEASE indication primitive optionally with RTTP APDU as user data

TABLE A-3/X.228 (Part 2 of 3)

Outgoing Event List

Abbreviated name	Source	Name and description
A-RELreq	ACSE	A-RELEASE request primitive
A-RELres	ACSE	A-RELEASE response primitive
P-ASreq	PS-provider	P-ACTIVITY-START request primitive
P-MSreq	PS-provider	P-MINOR-SYNCHRONIZE request primitive
P-MSres	PS-provider	P-MINOR-SYNCHRONIZE response primitive
P-AEreq	PS-provider	P-ACTIVITY-END request primitive
P-AEres	PS-provider	P-ACTIVITY-END response primitive
P-CGreq	PS-provider	P-CONTROL-GIVE request primitive
P-UEreq	PS-provider	P-U-EXCEPTION-REPORT request primitive
P-AIreq	PS-provider	P-ACTIVITY-INTERRUPT request primitive
P-AIres	PS-provider	P-ACTIVITY-INTERRUPT response primitive
P-ADreq	PS-provider	P-ACTIVITY-DISCARD request primitive
P-ADres	PS-provider	P-ACTIVITY-DISCARD response primitive
P-ARreq	PS-provider	P-ACTIVITY-RESUME request primitive

TABLE A-3/X.228 (Part 3 of 3)

Outgoing Event List

Abbreviated name	Source	Name and description
a-ab	RTPM	association aborted, recover
a-res	RTPM	activity resumption by the receiving RTPM
a-ret	RTPM	activity completed, discarded, or interrupted
ass-ab	RTPM	start of association-abort procedure
ass-rec	RTPM	start of association-recovery procedure
ass-rec-neg	RTPM	association-recovery unsuccessful
next	RTPM	transfer of RTTR APDU
p-ab	RTPM	start of provider-abort procedure
rt-ab	RTPM	RTAB received
tr-discard	RTPM	start of transfer-discard procedure
tr-interr	RTPM	start of transfer-interrupt procedure
tr-p-ab	RTPM	start of procedures transfer-abort followed by provider-abort
tr-pos	RTPM	transfer successful completed
tr-res	RTPM	start of transfer-resumption procedure
transfer	RTPM	start of transfer or transfer-retry procedures
u-exr	RTPM	start of user-exception-report procedure

TABLE A-4/X.228

Predicates

Code	Name and description
p1	RTPM can support the requested application-association
p2	Turn assigned to RTPM
p5	RTPM can support association-recovery
p6	transient rejection of association-recovery
p11	association-initiating RTPM
p30	only one RTTR APDU required to transfer the encoded-APDU-value (no checkpointing)
p31	RTTR APDU is the last one in a series of RTTR APDUs to transfer the encoded-APDU-value
p32	outstanding-minor-syncs < window-size
p33	outstanding-minor-syncs = 0
p34	sending RTPM is willing to recover from P-PEind
p35	checkpoint-confirmed (at least on P-MScnf received)
p361	reason parameter value of P-UEind is "receiving ability jeopardized"
p362	reason parameter value of P-UEind is "unrecoverable procedure error"
p363	reason parameter value of P-UEind is "non-specific error"
p364	reason parameter value of P-UEind is "sequence error"
p365	reason parameter value of P-UEind is "local SS-user error"
p41	received RTTR secured
p43	transfer to be resumed was already completed
p44	receiving RTPM is willing to perform and ignore transfer
p45	receiving RTPM can resume the activity
p46	receiving RTPM is willing to perform the association-abort procedure
p91	RTAB abort-reason field value is "user-error"
p92	RTAB abort-reason field value is "permanent-error"
p93	RTAB abort-reason field id value is "transfer-completed"

TABLE A-5/X.228

Specific Actions

Code	Name and description
a1	association-initiating RTPM = TRUE
a2	association-initiating RTPM = FALSE
a30	outstanding-minor-syncs = 0, set timer tr to transfer-time, checkpoint-confirmed = FALSE
a31	outstanding-minor-syncs = outstanding-minor-syncs +1
a32	outstanding-minor-syncs = outstanding-minor-syncs −1, checkpoint-confirmed = TRUE
a33	outstanding-minor-syncs = 0
a35	reset timer tr
a38	set timer rec to local recovery time
a39	reset timer rec
a41	set reason parameter value of P-UEreq to "sequence error"

TABLE A-6/X.228

RTPM State Table
Association-establishment

	STA0	STA01	STA02
RT-OPreq	p1: RTORQ [a1] STA01		
RTORQ	p1: RT-OPind [a2] STA02 ¬p1: RTORJ STA0		
RT-OPres+			p2: RTOAC STA21 ¬p2: RTOAC STA22
RT-OPres−			RTORJ STA0
RTOAC		p2: RT-OPcnf+ STA11 ¬p2: RT-OPcnf+ STA12	
RTORJ		RT-OPcnf− STA0	
A-ASCcnf−		RT-OPcnf− STA0	
A-PABind		RT-PAind STA0	RT-PAind STA0

TABLE A-7/X.228

RTPM State Table
Association Established, Outside Transfer

	STA11	STA12	STA21	STA22
RT-TRreq	transfer STA30		transfer STA30	
P-ASind		STA40		STA40
P-AIind		P-AIres STA12		P-AIres STA22
P-ARind		[a39] a-res STA45		[a39] a-res STA45
P-ADind		ass-ab STA48		ass-ab STA48
RT-TPreq		RTTP STA12		RTTP STA22
RTTP	RT-TPind STA11		RT-TPind STA21	
RT-TGreq	P-CGreq STA12		P-CGreq STA22	
P-CGind		RT-TGind STA11		RT-TGind STA21
RT-CLreq	A-RELreq STA92			
A-RELind				RT-CLind STA91
A-PABind	ass-rec STA51	ass-rec STA51	ass-rec STA52	ass-rec STA52
RT-UAreq	RTAB STA0	RTAB STA0	RTAB STA0	RTAB STA0
RTAB	rt-ab STA72	rt-ab STA72	rt-ab STA72	rt-ab STA72
rec-timeout		p-ab STA71		p-ab STA71

TABLE A-8/X.228 (Part 1 of 3)

RTPM State Table
Sending RTPM, Transfer

	STA30	STA31	STA32	STA321*
transfer	p30: [a30] P-ASreq RTTR P-AEreq STA32 ¬p30: [a30] P-ASreq next STA30			
next	p32&¬p31: RTTR P-MSreq [a31] next STA30 p32&31: RTTR P-AEreq STA32 ¬p32: STA31			
P-MScnf	[a32] STA30	[a32] next STA30	[a32] STA32	
P-AEcnf			p33: tr-pos STA321	
tr-pos				p11: [a35] RT-TRcnf+ STA11 ¬p11: [a35] RT-TRcnf+ STA21
tr-timeout	tr-discard [a38] STA35	tr-discard [a38] STA35	tr-discard [a38] STA35	

TABLE A-8/X.228 (Part 2 of 3)

RTPM State Table
Sending RTPM, Transfer

	STA30	STA31	STA32
P-UEind	p361: tr-p-ab STA71 p362: tr-discard STA36 p363: tr-discard STA35 p364: tr-discard STA34 p365&p35: tr-interr STA37 p365&¬p35: tr-discard STA36	p361: tr-p-ab STA71 p362: tr-discard STA36 p363: tr-discard STA35 p364: tr-discard STA34 p365&p35: tr-interr STA37 p365&¬p35: tr-discard STA36	p361: tr-p-ab STA71 p362: tr-discard STA36 p363: tr-discard STA35 p364: tr-discard STA34 p365&p35: tr-interr STA37 p365&¬p35: tr-discard STA36
P-PEind	p34&p35: tr-interr STA37 p34&¬p35: tr-discard STA36 ¬p34: tr-p-ab STA71	p34&p35: tr-interr STA37 p34&¬p35: tr-discard STA36 ¬p34: tr-p-ab STA71	p34&p35: tr-interr STA37 p34&¬p35: tr-discard STA36 ¬p34: tr-p-ab STA71

TABLE A-8/X.228 (Part 3 of 3)

RTPM State Table
Sending RTPM, Transfer

	STA30	STA31	STA32
s-problem-1	p35: tr-interr STA37 ¬p35: tr-discard STA36	p35: tr-interr STA37 ¬p35: tr-discard STA36	p35: tr-interr STA37 ¬p35: tr-discard STA36
s-problem-2	tr-discard STA36	tr-discard STA36	tr-discard STA36
s-problem-3	ass-ab STA38	ass-ab STA38	ass-ab STA38
A-PABind	a-ab STA381	a-ab STA381	a-ab STA381
RT-UAreq	RTAB STA0	RTAB STA0	RTAB STA0
RTAB	rt-ab STA39	rt-ab STA39	rt-ab STA39
RTTP	RT-TPind STA30	RT-TPind STA31	RT-TPind STA32

TABLE A-9/X.228

RTPM State Table
Sending RTPM, Error Handling

	STA34*	STA341	STA35*	STA351	STA36*	STA361
tr-discard	P-ADreq STA341		P-ADreq STA351		P-ADreq STA361	
P-ADcnf		tr-pos STA321		p11: [a35] RT-TRcnf– STA11 ¬p11: [a35] RT-TRcnf– STA21		transfer STA30
A-PABind		a-ab STA381		a-ab STA381		a-ab STA381
RT-UAreq		RTAB STA0		RTAB STA0		RTAB STA0
RTAB		rt-ab STA39		rt-ab STA39		rt-ab STA39
RTTP		RT-TPind STA341		RT-TPind STA351		RT-TPind STA361
tr-timeout		[a38] STA351		[a38] STA351		[a38] STA351
rec-timeout				tr-p-ab STA71		

TABLE A-10/X.228

RTPM State Table
Sending RTPM, Error Handling

	STA37*	STA371	STA372*
tr-interr	P-AIreq STA371		
P-AIcnf		tr-res STA372	
tr-res			p35: [a33] P-ARreq next STA30 ¬p35: P-ARreq tr-discard STA36
A-PABind		a-ab STA381	
RT-UAreq		RTAB STA0	
RTAB		rt-ab STA39	
RTTP		RT-TPind STA371	
tr-timeout		tr-p-ab STA71	

TABLE A-11/X.228

RTPM State Table
Sending RTPM, Error Handling

	STA38*	STA381*	STA39*
ass-ab	RTAB a-ab STA381		
a-ab		p11: ass-rec STA53 ¬p11: STA532	
rt-ab			p93&p11: RT-TRcnf+ ass-rec STA51 p93&¬p11 RT-TRcnf+ ass-rec STA52 p91: RT-TRcnf− RT-UAind STA0 p92: RT-TRcnf− RT-PAind STA0 ¬p91&¬p92: a-ab STA381

TABLE A-12/X.228

RTPM State Table
Receiving RTPM

	STA40	STA41	STA400	STA410	STA42
RTTR	STA41		STA 410		
P-MSind		p41: P-MSres STA40		P-MSres STA400	
P-AEind		RT-TRind P-AEres a-ret STA43		P-AEres a-ret STA43	
P-AIind	[a38] P-AIres a-ret STA43	[a38] P-AIres a-ret STA43	[a38] P-AIres a-ret STA43	[a38] P-AIres a-ret STA43	P-AIres a-ret STA43
P-ADind	P-ADres a-ret STA43	P-ADres a-ret STA43	P-ADres a-ret STA43	P-ADres a-ret STA43	[a39] P-ADres a-ret STA43
P-PEind	STA40	STA41	STA400	STA410	STA42
r-problem-1	u-exr STA44	u-exr STA44	u-exr STA44	u-exr STA44	
r-problem-2	ass-ab STA48	ass-ab STA48	ass-ab STA48	ass-ab STA48	ass-ab STA48
A-PABind	a-ab STA481	a-ab STA481	a-ab STA481	a-ab STA481	a-ab STA481
RT-TPreq	RTTP STA40	RTTP STA41	RTTP STA400	RTTP STA410	
RT-UAreq	RTAB STA0	RTAB STA0	RTAB STA0	RTAB STA0	RTAB STA0
RTAB	rt-ab STA49	rt-ab STA49	rt-ab STA49	rt-ab STA49	rt-ab STA49
rec-timeout					RT-PAind RTAB STA0

TABLE A-13/X.228 (Part 1 of 2)

RTPM State Table
Receiving RTPM Error Handling

	STA43*	STA44*	STA45*
a-ret	p11: STA12 ¬p11: STA22		
u-exr		P-UEreq: [a38] STA42	
a-res			¬p43&p45: STA40 p43&p44&p45: STA400 p43&¬p44&p45: [a41] u-exr STA44 ¬p45&¬p46: u-exr STA44 ¬p45&p46: ass-ab STA48

TABLE A-13/X.228 (Part 2 of 2)

RTPM State Table
Receiving RTPM Error Handling

	STA48*	STA481*	STA49*
ass-ab	RTAB a-ab STA481		
a-ab		p11: ass-rec STA54 ¬p11: ass-rec STA542	
rt-ab			p91: RT-UAind STA0 p92: RT-PAind STA0 ¬p91&¬p92: a-ab STA481

TABLE A-14/X.228 (Part 1 of 2)

RTPM State Table
Association-recovery Outside Transfer

	STA51*	STA510	STA52
ass-rec	p5: [a38] RTORQ STA510 ¬p5: p-ab STA70		[a38] STA52
RTORQ			p5&p2: [a39] RTOAC STA21 p5&¬p2: [a39] RTOAC STA22 ¬p5&p6: RTORJ STA52 ¬p5&¬p6: RTORJ p-ab STA70
RTOAC		p5&p2: [a39] STA11 p5&¬p2: [a39] STA12	

TABLE A-14/X.228 (Part 2 of 2)

RTPM State Table
Association-recovery Outside Transfer

	STA51*	STA510	STA52
RTORJ		ass-rec-neg STA51	
A-ASCcnf–		ass-rec-neg STA51	
A-PABind		ass-rec-neg STA51	
ass-rec-neg	p6: ass-rec STA51 ¬p6: p-ab STA70		
rec-timeout		p-ab STA71	p-ab STA70

TABLE A-15/X.228

RTPM State Table
Association-recovery During Transfer

	STA53*	STA531	STA532	STA54*	STA541	STA542
ass-rec	RTORQ STA531			[a38] RTORQ STA541		[a38] STA542
RTORQ			p5&p2: RTOAC tr-res STA372 ¬p5&p6: RTORJ STA532 ¬p5&¬p6: RTORJ tr-p-ab STA70			p5&¬p2: RTOAC [a39] STA22 ¬p5&p6: RTORJ STA542 ¬p5&¬p6: RTORJ p-ab STA70
RTOAC		tr-res STA372			[a39] STA12	
RTORJ		ass-rec-neg STA53			ass-rec-neg STA54	
A-ASCcnf–		ass-rec-neg STA53			ass-rec-neg STA54	
A-PABind		ass-rec-neg STA53			ass-rec-neg STA54	
ass-rec-neg	p6: ass-rec STA53 ¬p6: tr-p-ab STA70			p6: ass-rec STA54 ¬p6: p-ab STA70		
tr-timeout		tr-p-ab STA71	tr-p-ab STA70			
rec-timeout					p-ab STA71	p-ab STA70

TABLE A-16/X.228

RTPM State Table
Abort and Association-release

	STA70*	STA71*	STA72*	STA91	STA92
tr-p-ab	RT-TRcnf – RT-PAind STA0	RT-TRcnf – RTAB RT-PAind STA0			
p-ab	RT-PAind STA0	RT-PAind RTAB STA0			
rt-ab			p91: RT-UAind STA0 p92: RT-PAind STA0		
RT-CLres				A-RELres STA0	
A-RELcnf					RT-CLcnf STA0
A-PABind					p-ab STA70
RTAB					rt-ab STA72
RT-UAreq				RTAB STA0	

ANNEX B

(to Recommendation X.228)

Differences between this Recommendation and Recommendation X.410-1984

This annex is not part of this Recommendation.

This annex describes the technical differences between the protocol for Reliable Transfer of this Recommendation and the corresponding protocol of CCITT Recommendation X.410-1984.

In X.410-1984 mode this Recommendation and its use of ACSE and Presentation service is bit compatible to Recommendation X.410-1984 under consideration of the clarifications and errata of the X.400-series Implementors Guide V.5.

B.1 *Application protocol data units*

B.1.1 *PConnect*

1) The Set type and its two elements (Data Transfer Syntax and pUser Data) are now Presentation protocol control information (PPCI). The Elements of the RTORQapdu are the elements of the SETpUserData.
2) The application Protocol element is now OPTIONAL and solely used in X.41-1984 mode.
3) Implicit tagging of the SET in normal mode.

B.1.2 *PAccept*

1) The SET type and its two elements (DataTransferSyntax and pUserData) are now Presentation protocol control information. The elements of the RTOACapdu are the elements of the SET pUserData.
2) Implicit tagging of the SET in normal mode.

B.1.3 *PRefuse*

1) The SET type is now Presentation protocol control information. The elements of the RTORJapdu are the elements of the PRefuse SET.
2) Implicit tagging of the SET in normal mode.
3) Additional optional user data field in normal mode.

B.1.4 *Datatransfersyntax*

This information is now Presentation protocol control information.

B.1.5 *AbortInformation*

1) The SET type is now Presentation protocol control information. The elements of the RTABapdu are the elements of the AbortInformation SET.
2) Implicit tagging of the SET in normal mode.
3) Additional optional user data field in normal mode.

B.1.6 *AbortReason*

Add: Values (5) to (6) inclusive. Value (7) was added by the Addendum of the X.400 series Implementors Guide Version 5.

B.2 *Procedures and Mapping*

General Mapping onto used services

Change: From: onto Session Services

To: Mapping onto ACSE and Presentation services.

ANNEX C

(to Recommendation X.228)

Summary of Assigned Object Identifier Values

This annex is not an integral Part of this Recommendation.

This annex summarizes the object identifier values assigned in Recommendations X.218 and X.228.

```
{ joint-iso-ccitt reliable-transfer (3) apdus (0) }                 -- ASN.1 module defined in X.228

{ joint-iso-ccitt reliable-transfer (3) aseID (1) }                 -- RTSE identifier
                                                                    -- defined in X.228

{ joint-iso-ccitt transfert-fiable (3) abstract-syntax (2) }        -- Abstract syntax name
                                                                    -- defined in X.228
```

Recommendation X.229

REMOTE OPERATIONS: PROTOCOL SPECIFICATION[1]

(Melbourne, 1988)

The CCITT,

considering

(a) that Recommendation X.200 defines the Basic Reference Model of Open Systems Interconnection (OSI) for CCITT Applications;

(b) that Recommendation X.210 defines the service conventions for describing the services of the OSI reference model;

(c) that Recommendation X.216 defines the Presentation Layer service;

(d) that Recommendation X.217 defines the Association Control service;

(e) that Recommendation X.218 defines the Reliable Transfer service;

(f) that Recommendation X.219 defines the Remote Operations service and notation;

(g) that there is a need for common Remote Operations support for various applications,

unanimously declares

that this Recommendation defines the Remote Operations protocol of Open Systems Interconnection for CCITT Applications as given in the Scope and Field of Application.

CONTENTS

[1] Recommendation X.229 and ISO 9072-2 [Information processing systems – Text Communication – Remote Operations Part 2: Protocol specification] were developed in close collaboration and are technically aligned.

0 Introduction

This Recommendation specifies the protocol for the services provided by an application-service-element – the Remote Operations Service Element (ROSE) – to support interactive applications in a distributed open systems environment. This Recommendation is one of a set of Recommendations defining sets of application-service-elements commonly used by a number of applications.

Interactions between entities of a distributed application are modeled as Remote Operations, and defined using a Remote Operations Notation. A Remote Operation is requested by one entity; the other entity attempts to perform the Remote Operation and then reports the outcome of the attempt. Remote Operations are supported by the ROSE.

This Recommendation is technically aligned with ISO 9072-2.

1 Scope and field of application

This Recommendation specifies the protocol (abstract syntax) and procedures for the Remote Operation Service Element (Recommendation X.219). The ROSE services are provided in conjunction with the Association Control Service Element (ACSE) services (Recommendation X.217) and the ACSE protocol (Recommendation X.227), optionally the Reliable Transfer Service Element (RTSE) services (Recommendation X.218) and the RTSE protocol (Recommendation X.228), and the presentation-service (Recommendation X.216).

The ROSE procedures are defined in terms of:

a) the interactions between peer ROSE protocol machines through the use of RTSE services or the presentation-service;

b) the interactions between the ROSE protocol machine and its service-user.

This Recommendation specifies conformance requirements for systems implementing these procedures.

2 References

Recommendation X.200 – Reference model of open systems interconnection for CCITT applications (see also ISO 7498).

Recommendation X.208 – Specification of abstract syntax notation (see also ISO 8824).

Recommendation X.209 – Specification of basic encoding rules for the abstract syntax notation (see also ISO 8825).

Recommendation X.210 – Open systems interconnection layer service definition conventions (see also ISO/TR 8509).

Recommendation X.216 – Presentation service definition for open systems interconnection for CCITT applications (see also ISO 8822).

Recommendation X.217 – Association control service definition for CCITT applications (see also ISO 8649).

Recommendation X.218 – Reliable transfer: model and service definition (see also ISO 9066-1).

Recommendation X.219 – Remote operations: model, notation and service definition (see also ISO 9072-1).

Recommendation X.227 – Association control protocol specification for CCITT applications (see also ISO 8650).

Recommendation X.228 – Reliable transfer: protocol specfication (see also ISO 9066-2).

3 Definitions

3.1 *Reference model definitions*

This Recommendation is based on the concepts developed in Recommendation X.200 amd makes use of the following terms defined in it:

a) application layer;

b) application-process;

c) application-entity;

d) application-service-element;

e) application-protocol-data-unit;

f) application-protocol-control-information;
g) presentation-service;
h) presentation-connection;
i) session-service;
j) session-connection;
k) transfer syntax; and
l) user-element.

3.2 *Service conventions definitions*

This Recommendation makes use of the following terms defined in Recommendation X.210:

a) service-provider;
b) service-user;
c) confirmed service;
d) non-confirmed service;
e) provider-initiated service;
f) primitive;
g) request (primitive);
h) indication (primitive);
i) response (primitive); and
j) confirm (primitive).

3.3 *Presentation service definitions*

This Recommendation makes use of the following terms defined in Recommendation X.216:

a) abstract syntax;
b) abstract syntax name;
c) presentation context.

3.4 *Association control definitions*

This Recommendation makes use of the following terms defined in Recommendation X.217:

a) application-association; association;
b) application context;
c) association control service element.

3.4 *Reliable transfer definitions*

This Recommendation makes use of the following terms defined in Recommendation X.218:

a) reliable transfer service element.

3.6 *ROSE service definitions*

This Recommendation makes use of the following terms defined in Recommendation X.219:

a) association-initiating-application-entity; association-initiator;
b) association-responding-application-entity; association-responder;
c) invoking-application-entity; invoker;
d) performing-application-entity; performer;
e) requestor;
f) acceptor;
g) linked-operations;
h) parent-operation;
i) child-operation;
j) RO-notation;

k) remote operation service element;

l) ROSE-provider;

m) ROSE-user;

n) RTSE-user;

o) remote operations.

3.7 *Remote operation protocol specification definitions*

For the purpose of this Recommendation the following definitions apply:

3.7.1 **remote-operation-protocol-machine**:

The protocol machine for the remote operation service element specified in this Recommendation.

3.7.2 **requesting-remote-operation-protocol-machine**:

The remote-operation-protocol-machine whose service-user is the requestor of a particular remote operation service element service.

3.7.3 **accepting-remote-operation-protocol-machine**:

The remote-operation-protocol-machine whose service-user is the acceptor for a particular remote operation service element service.

4 Abbreviations

4.1 *Data units*

APDU	application-protocol-data-unit.

4.2 *Types of application-protocol-data-units*

The following abbreviations have been given to the application-protocol-data-units defined in this Recommendation.

ROIV	RO-INVOKE application-protocol-data-unit
RORS	RO-RESULT application-protocol-data-unit
ROER	RO-ERROR application-protocol-data-unit
RORJ	RO-REJECT application-protocol-data-unit

4.3 *Other abbreviations*

The following abbreviations are used in this Recommendation.

AE	application entiry
ACSE	association control service element
ASE	application service element
RO (or ROS)	remote operations
ROPM	remote operations protocol machine
ROSE	remote operations service element
RT	reliable transfer
RTSE	reliable transfer service element

5 Conventions

This Recommendation employs a tabular presentation of its APDU fields. In clause 7, tables are presented for each ROSE APDU. Each field is summarized using the following notation:

M	presence is mandatory
U	presence is a ROSE-user option
req	source is related request primitive
ind	sink is related indication primitive
resp	source is related response primitive
conf	sink is related confirm primitive
sp	source or sink is the ROPM

The structure of each ROSE APDU is specified in clause 9 using the abstract syntax notation of Recommendation X.208.

6 Overview of the protocol

6.1 *Service provision*

The protocol specified in Recommendation provides the ROSE services defined in Recommendation X.219. These services are listed in Table 1/X.229.

TABLE 1/X.229

ROSE services summary

Service	Type
RO-INVOKE	Non-confirmed
RO-RESULT	Non-confirmed
RO-ERROR	Non-confirmed
RO-REJECT-U	Non-confirmed
RO-REJECT-P	Provider-initiated

6.2 *Use of services*

The ROSE protocol specified in this Recommendation needs a transfer service to pass information in the form of ROSE APDUs between peer application-entities (AEs).

Two transfer services may be used alternatively:

a) the RTSE services, if the RTSE is included in the application-context; or

b) the presentation-service, if the RTSE is not included in the application-context.

In both cases, an existing application-association, established and released by means of the ACSE services, is assumed.

6.2.1 *Use of the RTSE services*

If the RTSE is included in the application-context, this Recommendation assumes that the ROPM is the sole user of the RT-TRANSFER service and the RT-TURN-GIVE service.

The initiating AE may only request the release of the application-association by means of the RT-CLOSE service if it possesses the turn. Therefore the RTSE-user and the ROPM are the user of the RT-TURN-PLEASE service.

The ROPM is the user of the RT-U-ABORT and RT-P-ABORT services.

6.2.2 *Use of the presentation-service*

If the RTSE is not included in the application context, the ROPM is a user of the P-DATA service.

6.3 *Model*

The remote-operation-protocol-machine (ROPM) communicates with its service-user by means of primitives defined in Recommendation X.219. Each invocation of the ROPM controls a single application-association.

The ROPM is driven by ROSE service request primitives from its service-user, and by indication and confirm primitives of the RTSE services, or the presentation-service. The ROPM, in turn, issues indication primitives to its service-user, and request primitives on the used RTSE services, or the presentation-service. If the RTSE is included in the application-context, the RT-TRANSFER indication, RT-TRANSFER request and RT-TRANSFER confirm primitives are used. In the case of an application-context excluding RTSE, the presentation-service P-DATA request, and P-DATA indication primitives are used. In this case the transfer is not confirmed.

The reception of an ROSE service primitive, or of an RTSE service or of a presentation-service primitive, and the generation of dependent actions are considered to be individual.

During the exchange of APDUs, the existence of both, the association-initiating AE and the association-responding AE is presumed. How these AEs are created is beyond the scope of this Recommendation.

During the execution of operations, the existence of an application-association between the peer AEs is presumed. How this application-association is established and released is beyond the scope of this Recommendation (see Recommendations X.219, X.217, X.227, X.218 and X.228).

Note – Each application-association may be identified in an end system by an internal, implementation dependent mechanism so that the ROSE service-user and the ROPM can refer to it.

7 **Elements of procedure**

The ROSE protocol consists of the following elements of procedure:

a) invocation;

b) return-result;

c) return-error;

d) user-reject;

e) provider-reject.

In the following clauses, a summary of each of these elements of procedure is presented. This consists of a summary of the relevant APDUs, and high-level overview of the relationship between the ROSE service primitives, the APDUs involved, and the transfer service that is used.

The generic terms transfer service, transfer service-provider, transfer request, and transfer indication are used in the context of clause 7. Clause 8 describes how these generic service primitives are mapped either on to the RTSE services or the presentation-service.

In clause 9 a detailed specification of the ROSE APDUs is given using the notation defined in Recommendation X.208.

7.1 *Invocation*

7.1.1 *Purpose*

The invocation procedure is used by one AE (the invoker) to request an operation to be performed by the other AE (the performer).

7.1.2 *APDUs used*

The invocation procedure uses the RO-INVOKER (ROIV) APDU.

The fields of the ROIV APDU are listed in Table 2/X.229.

TABLE 2/X.229

ROIV APDU fields

Field name	Presence	Source	Sink
Invoke-ID	M	req	ind
Linked-ID	U	req	ind
Operation-value	M	req	ind
Argument	U	req	ind

7.1.3 *Invocation procedure*

This procedure is driven by the following events:

a) an RO-INVOKE request primitive from the requestor;

b) an ROIV APDU as user-data of a transfer indication primitive.

7.1.3.1 *RO-INVOKE request primitive*

The requesting ROPM forms an ROIV APDU from the parameter values of the RO-INVOKE request primitive. It issues a transfer request primitive. The user-data parameter of the transfer request primitive contains the ROIV APDU.

The requesting ROPM waits either for a transfer indication primitive from the transfer service-provider or any other primitive from the requestor.

7.1.3.2 *ROIV APDU*

The accepting ROPM receives an ROIV APDU from its peer as user-data on a transfer indication primitive. If any of the fields of the ROIV APDU are unacceptable to this ROPM, the provider-reject procedure is performed, and no RO-INVOKE indication primitive is issued by the ROPM.

If the ROIV APDU is acceptable to the accepting ROPM, it issues an RO-INVOKE indication primitive to the acceptor. The RO-INVOKE indication primitive parameters are derived from the ROIV APDU.

The accepting ROPM waits either for a transfer indication primitive from the transfer service-provider or any other primitive from the acceptor.

7.1.4 *Use of the ROIV APDU fields*

The ROIV fields are used as follows.

7.1.4.1 *Invoke-ID*

This is the Invoke-ID parameter value of the RO-INVOKE request primitive. It appears as the Invoke-ID parameter value of the RO-INVOKE indication primitive.

The value of this field is transparent to the ROPM, however the value may be used in the provider reject procedure.

7.1.4.2 *Linked-ID*

This is the Linked-ID parameter value of the RO-INVOKE request primitive. It appears as the Linked-ID parameter value of the RO-INVOKE indication primitive.

The value of this field is transparent to the ROPM.

7.1.4.3 *Operation-value*

This is the Operation-value parameter value of the RO-INVOKE request primitive. It appears as the Operation-value parameter value of the RO-INVOKE indication primitive.

The value of this field is transparent to the ROPM.

7.1.4.4 *Argument*

This is the Argument parameter value of the RO-INVOKE request primitive. It appears as the Argument parameter value of the RO-INVOKE indication primitive.

The value of this field is transparent to the ROPM.

7.2 *Return-result*

7.2.1 *Purpose*

The return-result procedure is used by one AE (the performer) to request the transfer of the result of a successfully performed operation to the other AE (the invoker).

7.2.2 *APDUs used*

The return-result procedure uses the RO-RESULT (RORS) APDU.

The fields of the RORS APDU are listed in Table 3/X.229.

TABLE 3/X.229

RORS APDU fields

Field name	Presence	Source	Sink
Invoke-ID	M	req	ind
Operation-value	U	req	ind
Result	U	req	ind

7.2.3 *Return-result procedure*

This procedure is driven by the following events:

a) an RO-RESULT request primitive from the requestor;

b) an RORS APDU are user-data of a transfer indication primitive.

7.2.3.1 *RO-RESULT request primitive*

The requesting ROPM forms an RORS APDU from the parameter values of the RO-RESULT request primitive. It issues a transfer request primitive. The user-data parameter of the transfer request primitive contains the RORS APDU.

The requesting ROPM waits either for a transfer indication primitive from the transfer service-provider or any other primitive from the requestor.

7.2.3.2 *RORS APDU*

The accepting ROPM receives an RORS APDU from its peer as user-data on a transfer indication primitive. If any of the fields of the RORS APDU are unacceptable to this ROPM, the provider-reject procedure is performed, and no RO-RESULT indication primitive is issued by the ROPM.

If the RORS APDU is acceptable to the accepting ROPM, it issues an RO-RESULT indication primitive to the acceptor. The RO-RESULT indication primitive parameters are derived from the RORS APDU.

The accepting ROPM waits either for a transfer primitive from the transfer service-provider or any other primitive from the acceptor.

7.2.4 *Use of the RORS APDU fields*

The RORS fields are used as follows.

7.2.4.1 *Invoke-ID*

This is the Invoke-ID parameter value of the RO-RESULT request primitive. It appears as the Invoke-ID parameter value of the RO-RESULT indication primitive.

The value of this field is transparent to the ROPM, however the value may be used in the provider-reject procedure.

7.2.4.2 *Operation-value*

This is the Operation-value parameter value of the RO-RESULT request primitive. It appears as the Operation-value parameter value of the RO-RESULT indication primitive.

The value of this field is transparent to the ROPM.

This field shall be present only if the result field is present.

7.2.4.3 *Result*

This is the Result parameter value of the RO-RESULT request primitive. It appears as the Result parameter value of the RO-RESULT indication primitive.

The value of this field is transparent to the ROPM.

7.3 *Return-error*

7.3.1 *Purpose*

The return-error procedure is used by one AE (the performer) to request the transfer of the error information in the case of an unsuccessfully performed operation to the other AE (the invoker).

7.3.2 *APDUs used*

The return-error procedure uses the RO-ERROR (ROER) APDU.

The fields of the ROER APDU are listed in Table 4/X.229.

TABLE 4/X.229

ROER APDU fields

Field name	Presence	Source	Sink
Invoke-ID	M	req	ind
Error-value	M	req	ind
Error-parameter	U	req	ind

7.3.3 *Return-error procedure*

This procedure is driven by the following events:

a) an RO-ERROR request primitive from the requestor;

b) an ROER APDU as user-data of a transfer indication primitive.

7.3.3.1 *RO-ERROR request primitive*

The requesting ROPM forms an ROER APDU from the parameter values of the RO-ERROR request primitive. It issues a transfer request primitive. The user-data parameter of the transfer request primitive contains the ROER APDU.

The requesting ROPM waits either for a transfer primitive from the transfer service-provider or any other primitive from the requestor.

7.3.3.2 *ROER APDU*

The accepting ROPM receives an ROER APDU from its peer as user-data on a transfer indication primitive. If any of the fields of the ROER APDU are unacceptable to this ROPM, the provider-reject procedure is performed, and no RO-ERROR indication primitive is issued by the ROPM.

If the ROER APDU is acceptable to the accepting ROPM, it issues an RO-ERROR indication primitive to the acceptor. The RO-ERROR indication primitive parameters are derived from the ROER APDU.

The accepting ROPM waits either for a transfer indication primitive from the transfer service-provider or any other primitive from the acceptor.

7.3.4 *Use of the ROER APDU fields*

The ROER fields are used as follows.

7.3.4.1 *Invoke-ID*

This is the Invoke-ID parameter value of the RO-ERROR request primitive. It appears as the Invoke-ID parameter value of the RO-ERROR indication primitive.

The value of this field is transparent to the ROPM, however the value may be used in the provider-reject procedure.

7.3.4.2 *Error-value*

This is the Error-value parameter value of the RO-ERROR request primitive. It appears as the Error-value parameter value of the RO-ERROR indication primitive.

The value of this field is transparent to the ROPM.

7.3.4.3 *Error-parameter*

This is the Error-parameter parameter value of the RO-ERROR request primitive. It appears as the Error-parameter parameter value of the RO-ERROR indication primitive.

The value of this field is transparent to the ROPM.

7.4 *User-reject*

7.4.1 *Purpose*

The user-reject procedure is used by one AE to reject the request (invocation) or reply (result or error) of the other AE.

7.4.2 *APDUs used*

The user-reject procedure uses the RO-REJECT (RORJ) APDU. This RORJ APDU is used in addition by the provider-reject procedure.

The fields of the RORJ APDU used for the user-reject procedure are listed in Table 5/X.229.

TABLE 5/X.229

RORJ APDU fields used for user-reject

Field name	Presence	Source	Sink
Invoke-ID	M	req	ind
Problem (choice of): Invoke-problem Return-result-problem Return-error-problem	M	req	ind

7.4.3 *User-reject procedure*

This procedure is driven by the following events:

a) an RO-REJECT-U request primitive from the requestor;

b) an RORJ APDU as user-data of a transfer indication primitive.

7.4.3.1 *RO-REJECT-U request primitive*

The requesting ROPM forms an RORJ APDU from the parameter values of the RO-REJECT-U request primitive. It issues a transfer request primitive. The user-data parameter of the transfer request primitive contains the RORJ APDU.

The requesting ROPM waits either for a transfer indication primitive from the transfer service-provider or any other primitive from the acceptor.

7.4.3.2 *RORJ APDU*

The accepting ROPM receives an RORJ APDU from its peer as user-data on a transfer indication primitive. If any of the fields of the RORJ APDU are unacceptable to this ROPM, no RO-REJECT-U indication primitive is issued by the ROPM.

If the RORJ APDU is acceptable to the accepting ROPM and the fields of the RORJ APDU indicates a user reject (i.e. Invoke-problem, Return-result-problem, or Return-error-problem), it issues an RO-REJECT-U indication primitive to the acceptor. The RO-REJECT-U indication primitive parameters (Invoke-ID and Reject-reason) are derived from the RORJ APDU.

The accepting ROPM waits either for a transfer indication primitive from the transfer service-provider or any other primitive from the acceptor.

7.4.4 *Use of the RORJ APDU fields*

The RORJ fields are used as follows.

7.4.4.1 *Invoke-ID*

This is the Invoke-ID parameter value of the RO-REJECT-U request primitive. It appears as the Invoke-ID parameter value of the RO-REJECT-U indication primitive.

The value of this field is transparent to the ROPM.

7.4.4.2 *Problem*

This is the Problem parameter value of the RO-REJECT-U request primitive. It appears as the Problem parameter value of the RO-REJECT-U indication primitive.

The values used by the user-reject procedure are:

a) *Invoke problem*: user-reject of an RO-INVOKE indication primitive with values:

- duplicate-invocation:
  signifies that the Invoke-ID parameter violates the assignment rules of Recommendation X.219.
- unrecognized-operation:
  signifies that the operation is not one of those agreed between the ROSE-users.
- mistyped-argument:
  signifies that the type of the operation argument supplied is not that agreed between the ROSE-users
- resource-limitation:
  the performing ROSE-user is not able to perform the invoked operation due to resource limitation.
- initiator-releasing:
  the association-initiator is not willing to perform the invoked operation because it is about to attempt to release the application-association.
- unrecognized-linked-ID:
  signifies that there is no operation in progress with an Invoke-ID equal to the specified Linked-ID.
- linked-response-unexpected:
  signifies that the invoked operation referred to by the Link-ID is not a parent-operation.
- unexpected-child-operation:
  signifies that the invoked child-operation is not one that the invoked parent-operation referred to by the Linked-ID allows.

b) *Return-result-problem*: user-reject of an RO-RESULT indication primitive with values:

- unrecognized-invocation:
  signifies that no operation with the specified Invoke-ID is in progress.
- result-response-unexpected:
  signifies that the invoked operation does not report a result.
- mistyped-result:
  signifies that the type of the Result parameter supplied is not that agreed between the ROSE-users.

c) *Return-error-problem*: user-reject of an RO-ERROR indication primitive with values:

- unrecognized-invocation:
  signifies that no operation with the specified Invoke-ID is in progress
- error-response-unexpected:
  signifies that the invoked operation does not report failure
- unrecognized-error:
  signifies that the reported error is not one of those agreed between the ROSE-users

- unexpected-error:
  signifies that the reported error is not one that the invoked operation may report
- mistyped-parameter:
  signifies that the type of the error parameter supplied is not that agreed between the ROSE-users.

7.5 *Provider-reject*

7.5.1 *Purpose*

The provider-reject procedure is used to inform the ROSE user and the peer ROPM, if an ROPM detects a problem.

7.5.2 *APDUs used*

The provider-reject procedure uses the RO-REJECT (RORJ) APDU. This RORJ APDU is used in addition by the user-reject procedure.

The fields of the RORJ APDU used for the provider-reject procedure are listed in Table 6/X.229.

TABLE 6/X.229

RORJ APDU fields used for provider-reject

Field name	Presence	Source	Sink
Invoke-ID	M	sp	ind
Problem (choice of):	M	sp	ind
General-problem			

7.5.3 *Provider-reject procedure*

This procedure is driven by the following events:

a) an unacceptable APDU as user-data of a transfer indication primitive;

b) an RORJ APDU with the Problem parameter choice General-problem as user-data of a transfer indication primitive;

c) unsuccessful APDU transfer (e.g. association abort).

7.5.3.1 *Unacceptable APDU*

The receiving ROPM receives an APDU from its peer as user data on a transfer indication primitive. If any of the fields of the APDU (except RORJ APDU) are unacceptable to this ROPM, it forms an RORJ APDU with the Problem field choice General-problem and the Invoke-ID of the rejected APDU. The receiving ROPM issues a transfer request primitive. The user-data parameter of the transfer request primitive contains the RORJ APDU.

If the received unacceptable APDU is an RORJ APDU no new RORJ APDU is formed and transferred. In this case, or after the rejection of a locally specified number of APDUs, the application-association is released abnormally.

If the application-association is not released abnormally, the receiving ROPM waits either for a transfer indication primitive from the transfer service-provider or any other primitive from the requestor.

7.5.3.2 *RORJ APDU*

The receiving ROPM receives an RORJ APDU from its peer as user-data on a transfer indication primitive. If any of the fields of the RORJ APDU are unacceptable to this ROPM, the provider-reject procedure for an unacceptable APDU is performed.

If the RORJ APDU is acceptable to the accepting ROPM and the Problem field of the RORJ APDU indicates a General-problem, it issues an RO-REJECT-P indication primitive to the acceptor. The RO-REJECT-P indication primitive parameters (Invoke-ID and Reject-reason) are derived from the RORJ APDU.

The receiving ROPM waits either for a transfer indication primitive from the transfer service-provider or any other primitive from the acceptor.

7.5.3.3 *Unsuccessful APDU transfer*

If a sending ROPM is not able to transfer an APDU by means of the transfer request primitive (e.g. in the case of abnormal association release), the sending ROPM issues an RO-REJECT-P indication primitive to the requestor for each APDU not yet transferred.

The RO-REJECT-P indication primitive parameter Returned-parameters contains the parameters of the RO-INVOKE request, RO-RESULT request, RO-ERROR request or RO-REJECT-U request primitives.

After all Returned-parameters of the APDUs not transferred have been issued to the requestor, the application-association, if it still exists, is released abnormally.

7.5.4 *Use of the RORJ APDU fields*

The RORJ APDU fields are used as follows.

7.5.4.1 *Invoke-ID*

This is the Invoke-ID field of a rejected APDU and the Invoke-ID parameter of the RO-REJECT-P indication primitive. The type and value of this field may be NULL, if the Invoke-ID field of the rejected APDU is not detectable. In this case, the Invoke-ID parameter of the RO-REJECT-P indication primitive is omitted.

7.5.4.2 *Problem: General-problem*

This is the Problem parameter value of the RO-REJECT-P indication primitive. The values used by the provider-reject procedure are:

d) *General-problem*: provider-reject of an APDU with values:

- unrecognized-APDU:
  signifies that the type of the APDU, as evidenced by its Type Identifier, is not one of the four defined by this Recommendation.
- mistyped-APDU:
  signifies that the structure of the APDU does not conform to this Recommendation.
- badly-structured-APDU:
  signifies that the structure of the APDU does not conform to the standard notation and encoding, defined in Recommendations X.208 and X.209.

8 Mapping to used services

This clause defines how an ROPM transfers APDUs by means of:

a) the RTSE services, or

b) the presentation-service.

Clause 8.1 defines the mapping on the RTSE services, and clause 8.2 defines the mapping on the presentation-service.

Identification of the named abstract syntax in use is assumed for all ROSE services and is mapped onto used services, however this is a local matter and outside the scope of this Recommendation.

8.1 *Mapping on the RTSE services*

This clause defines how RTSE service primitives described in Recommendation X.218 are used by the ROPM. Table 7/X.229 defines the mapping of the ROSE service primitives and APDUs to the RTSE service primitives.

TABLE 7/X.229

RTSE mapping overview

ROSE service	ADPU	RTSE service
RO-INVOKE request/indication	ROIV	RT-TRANSFER request/indication/confirm
RO-RESULT request/indication	RORS	RT-TRANSFER request/indication/confirm
RO-ERROR request/indication	ROER	RT-TRANSFER request/indication/confirm
RO-REJECT-U request/indication	RORJ	RT-TRANSFER request/indication/confirm
RO-REJECT-P indication	RORJ	RT-TRANSFER request/indication/confirm
Managing the Turn	-	RT-TURN-PLEASE request/indication
	-	RT-TURN-GIVE request/indication

8.1.1 *Managing the turn*

A ROPM shall possess the turn before it can use the RT-TRANSFER service. The ROPM without the turn may issue a RT-TURN-PLEASE request primitive the priority parameter of which reflects the highest priority APDU awaiting transfer.

The ROPM which has the turn, may issue an RT-TURN-GIVE request primitive when it has no further APDUs to transfer. It will issue an RT-TURN-GIVE request primitive in response to an RT-TURN-PLEASE indication when it has no further APDUs to transfer of priority equal to or higher than that indicated in the RT-TURN-PLEASE indication primitive. If it has APDUs of lower priority still to transfer, it may issue an RT-TURN-PLEASE request whose priority reflects the highest priority APDU remaining to be transferred.

8.1.1.1 *Use of the RT-TURN-PLEASE service*

The ROPM issues the RT-TURN-PLEASE request primitive to request the turn. It may do so only if it does not already possess the turn. The RT-TURN-PLEASE service is a non-confirmed service.

The use of the RT-TURN-PLEASE service parameters is as follows:

Priority: this reflects the highest priority APDU awaiting transfer.

8.1.1.2 *Use of the RT-TURN-GIVE service*

The ROPM issues the RT-TURN-GIVE request primitive to relinquish the turn to its peer. It may do so only if it possesses the turn. The RT-TURN-GIVE service is a non-confirmed service with no parameters.

8.1.2 *APDU transfer*

Each APDU is transferred as user-data of the RT-TRANSFER service. The ROPM only issues an RT-TRANSFER request primitive, if the ROPM possesses the turn, and if there is no outstanding RT-TRANSFER confirm primitive.

8.1.2.1 *Use of the RT-TRANSFER service*

The RT-TRANSFER service is a confirmed service.

The use of the RT-TRANSFER request primitive parameters is as follows:

APDU
The APDU to be transferred. Its maximum size is not restricted in this mapping.

Transfer-time
This is specified by a local rule of the sending ROPM. It may be related to the priority of the APDU.

The use of the RT-TRANSFER indication primitive parameters is as follows:

APDU
The APDU transferred. Its maximum size is not restricted in this mapping.

The use of the RT-TRANSFER confirm primitive parameters is as follows:

APDU
The APDU not transferred within the Transfer-time. This parameter is only provided, if the value of the Result parameter is "APDU-not-transferred". In this case the ROPM issues a RO-REJECT-P indication primitive with the parameter Returned-parameters.

Result
The parameter value "APDU-transferred" indicates a positive confirm, the parameter value "APDU-not-transferred" indicates a negative confirm.

8.2 *Mapping on the presentation-service*

This clause defines how the presentation-service primitives described in Recommendation X.216 are used by the ROPM. Table 8/X.229 defines the mapping of the ROSE service primitives and APDUs to the presentation-service primitives.

TABLE 8/X.229

Presentation-service mapping overview

ROSE service	ADPU	Presentation service
RO-INVOKE request/indication	ROIV	P-DATA request/indication
RO-RESULT request/indication	RORS	P-DATA request/indication
RO-ERROR request/indication	ROER	P-DATA request/indication
RO-REJECT-U request/indication	RORJ	P-DATA request/indication
RO-REJECT-P request/indication	RORJ	P-DATA request/indication

8.2.1 *APDU transfer*

Each APDU is transferred as user-data of the P-DATA service.

8.2.1.1 *Use of the P-DATA service*

The P-DATA service is a non-confirmed service.

The use of the P-DATA request and P-DATA indication primitive parameters is as follows:

User Data
The APDU to be transferred. Its maximum size is not restricted in this mapping.

9 Abstract syntax definition of APDUs

The abstract syntax of each ROSE APDU is specified in this clause using the abstract syntax notation of Recommendation X.208 and is shown in Figure 1/X.229.

```
Remote-Operations-APDUs {joint-iso-ccitt remote-operations(4) apdus(1)}

DEFINITIONS ::=

BEGIN

EXPORTS rOSE, InvokeIDType;

-- the following macros are used as defined in Figure 4/X.219

IMPORTS   OPERATION, ERROR FROM Remote-Operation-Notation  {joint-iso-ccitt remote-operations(4)
                                                            notation(0)}

          APPLICATION-SERVICE-ELEMENT FROM  Remote-Operations-Notation-extension
                                            {joint-iso-ccitt remote-operations(4)
                                            notation-extension(2)};

rOSE  APPLICATION-SERVICE-ELEMENT ::= {joint-iso-ccitt remote-operations(4) aseID(3)}

-- APDUs
-- Types and values of operations and errors are defined in an ROSE-user protocol specification using
-- the RO-notation. Operation values are either of object identifier type or of integer type. If integer types are
-- used they shall be distinct within an abstract syntax. Error values are either of object identifier type
-- or integer types. If integer types are used they shall be distinct within an abstract syntax. There is no object
-- identifier specified for the abstrac syntax name for ROSE. However all ASN.1 data types defined in this
-- module shall be included in the named abstract syntax defined in the ROSE-user protocol specification.

ROSEapdus          ::=    CHOICE {
                              roiv-apdu [1] IMPLICIT ROIVapdu,
                              rors-apdu [2] IMPLICIT RORSapdu,
                              roer-apdu [3] IMPLICIT ROERapdu,
                              rorj-apdu [4] IMPLICIT RORJapdu}

-- ROSE protocol specification continued
```

FIGURE 1/X.229 (Sheet 1 of 3)

Abstract syntax specification of ROSE protocol

```
-- ROSE protocol specification continued


-- APDU types

ROIVapdu        ::=     SEQUENCE {
                              invokeID InvokeIDType,
                              linked-ID [0] IMPLICIT invokedIDType OPTIONAL,
                              operation-value OPERATION,
                              argument ANY DEFINED BY operation-value OPTIONAL}
                              -- ANY is filled by the single ASN.1 data type following
                              -- the key word ARGUMENT in the type definition
                              -- of a particular operation.


InvokeIDType    ::=     INTEGER


RORSapdu        ::=     SEQUENCE {
                              InvokeID invokeIDType,
                              SEQUENCE {operation-value OPERATION,
                                        result ANY DEFINED BY operation-value
                                        -- ANY is filled by the single ASN.1 data type
                                        -- following the key word RESULT in the type
                                        -- definition of a particular operation.
                                  } OPTIONAL }


ROERapdu        ::=     SEQUENCE {
                              InvokedID InvokedIDType,
                              error-value ERROR,
                              parameter ANY DEFINED BY error-value OPTIONAL}
                              -- ANY is filled by the single ASN.1 data type following
                              -- the key word PARAMETER in the type definition
                              -- of a particular operation.


RORJapdu        ::=     SEQUENCE {
                              InvokeID CHOICE {InvokeIDType, NULL},
                              problem CHOICE {
                                    [0] IMPLICIT GeneralProblem,
                                    [1] IMPLICIT InvokeProblem,
                                    [2] IMPLICIT ReturnResultProblem,
                                    [3] IMPLICIT ReturnErrorProblem}}


-- ROSE protocol specification continued
```

FIGURE 1/X.229 (Sheet 2 of 3)

Abstract syntax specification of ROSE protocol

```
-- ROSE protocol specification continued

GeneralProblem          ::=   INTEGER {               -- ROSE-provider detected
                                  unrecognisedAPDU(0),
                                  mistypedAPDU(1),
                                  badlyStructuredAPDU(2)}

InvokeProblem           ::=   INTEGER {               -- ROSE-user detected
                                  duplicateInvocation(0),
                                  unrecognisedOperation(1),
                                  mistypedArgument(2),
                                  resourceLimitation(3),
                                  initiatorReleasing(4),
                                  unrecognisedLinkedID(5),
                                  linkedResponseUnexpected(6),
                                  unexpectedChildOperation(7)}

ReturnResultProblem     ::=   INTEGER {               -- ROSE-user detected
                                  unrecognisedInvocation(0),
                                  resultResponseUnexpected(1),
                                  mistypedResult(2)}

ReturnErrorProblem      ::=   INTEGER {               -- ROSE-user detected
                                  unrecognisedInvocation(0),
                                  errorResponseUnexpected(1),
                                  unrecognisedError(2),
                                  unexpectedError(3),
                                  mistypedParameter(4)}

END     -- of ROSE protocol specification
```

FIGURE 1/X.229 (Sheet 3 of 3)

Abstract syntax specification of ROSE protocol

10 Conformance

An implementation claiming conformance to this Recommendation shall comply with the requirements in clauses 10.1 through 10.3.

10.1 *Statement requirements*

An implementor shall state the following:

a) the application context for which conformance is claimed, including whether the system supports the mapping of ROSE onto RTSE, onto the presentation-service, or both.

10.2 *Static requirements*

The system shall:

a) conform to the abstract syntax definition of APDUs defined in clause 9.

10.3 *Dynamic requirements*

The system shall:

a) conform to the elements of procedure defined in clause 7,

b) conform to the mappings to the used services, for which conformance is claimed, as defined in clause 8.

ANNEX A

(to Recommendation X.229)

ROPM state tables

This Annex forms an integral Part of this Recommendation.

A.1 *General*

This Annex defines a single Remote Operations Protocol Machine (ROPM) in terms of a state table. The state table shows the interrelationship between the state of an application-association, the incoming events that occur in the protocol, the actions taken, and, finally the resultant state of the application-association.

The ROPM state table does not constitute a formal definition of the ROPM. It is included to provide a more precise specification of the elements of procedure defined in clauses 7 and 8.

This Annex contains the following tables:

a) Table A-1/X.229 specifies the abbreviated name, source, and name/description of each incoming event. The sources are:
   1) ROSE-user (ROSE-user);
   2) peer ROPM (ROPM-peer);
   3) ROPM excluding the transfer part (ROPM);
   4) transfer part of the ROPM (ROPM-TR);
   5) either Presentation service-provider (PS-provider) and the Association Control Service Element (ACSE), or the Reliable Transfer Service Element (RTSE).

b) Table A-2/X.229 specifies the abbreviated name of each state of the ROPM.

c) Table A-3/X.229 specifies the abbreviated name of each state of the ROPM-TR.

d) Table A-4/X.229 specifies the abbreviated name, target, and name/description of each outgoing event. The targets are:
   1) ROSE-user (ROSE-user);
   2) peer ROPM (RPOM peer);
   3) ROPM excluding the transfer part (ROPM);
   4) transfer part of the ROPM (ROPM-TR); and
   5) either Presentation service-provider (PS-provider) and the Association Control Service Element (ACSE), or the Reliable Transfer Service Element (RTSE).

e) Table A-5/X.229 specifies the predicates.

f) Table A-6/X.229 specifies the ROPM state table using the abbreviations of the above tables.

g) Table A-7/X.229 specifies the ROPM-TR state table using the abbreviations of the above tables, if the RTSE is included in the application context.

h) Table A-8/X.229 specifies the ROPM-TR state table using the abbreviations of the above tables, if the RTSE is not included in the application context.

A.2 *Conventions*

The intersection of an incoming event (row) and a state (column) forms a cell.

In the state table, a blank cell represents the combination of an incoming event and a state that is not defined for the ROPM. (See A.3.1.)

A non-blank cell represents an incoming event and a state that is defined for the ROPM. Such a cell contains one or more action lists. An action list may be either mandatory or conditional. If a cell contains a mandatory action list, it is the only action list in the cell.

A mandatory action list contains:

a) optionally one or more outgoing events, and

b) a resultant state.

A conditional action list contains:

a) a predicate expression comprising predicates and Boolean operators (¬ represents the Boolean NOT), and

b) a mandatory action list (this mandatory action list is used only if the predicate expression is true).

A.3 *Actions to be taken by the ROPM*

The ROPM state table defines the action to be taken by the ROPM in terms of an optional outgoing event and the resultant state of the application-association.

A.3.1 *Invalid intersections*

Blank cells indicate an invalid intersection of an incoming event and state. If such an intersection occurs, one of the following actions is taken:

a) If the incoming event comes from the ROSE-user, any action taken by the ROPM is a local matter.

b) If the incoming event is related to a received APDU, PS-provider, ACSE, or RTSE; either the ROPM issues an AA-ABreq event to the ROPM-TR, or the ROPM-TR issues an ABORTreq to either the RTSE or ACSE and an AA-ABind to the ROPM.

A.3.2 *Valid intersections*

If the intersection of the state and incoming event is valid, one of the following actions is taken:

a) If the cell contains a mandatory action list, the ROPM takes the action specified.

b) If a cell contains one or more conditional action lists, for each predicate expression that is true, the ROPM takes the actions specified. If none of the predicate expressions are true, the ROPM takes one of the actions defined in § A.3.1.

TABLE A-1/X.229

Incoming event list

Abbreviated name	Source	Name and description
AA-ESTAB	RTSE ACSE	positive RT-OPEN response primitive or positive RT-OPEN confirm primitive positive A-ASSOCIATE response primitive or positive A-ASSOCIATE confirm primitive
RO-INVreq	ROSE-user	RO-INVOKE request primitive
RO-RESreq	ROSE-user	RO-RESULT request primitive
RO-ERRreq	ROSE-user	RO-ERROR request primitive
RO-RJUreq	ROSE-user	RO-REJECT-U request primitive
ROIV	ROPM-peer	valid RO-INVOKE APDU as user data on a TRANSind event
RORS	ROPM-peer	valid RO-RESULT APDU as user data on a TRANSind event
ROER	ROPM-peer	valid RO-ERROR APDU as user data on a TRANSind event
RORJu	ROPM-peer	valid RO-REJECT APDU (user-reject) as user data on a TRANSind event
RORJp	ROPM-peer	valid RO-REJECT APDU (provider-reject with General-problem) as user data on a TRANSind event
APDUua	ROPM-peer	unacceptable APDU as user data on a TRANSind event
TRANSind	ROPM-TR	transfer indication of an APDU
TRANSreq	ROPM	transfer request for an APDU
P-DATAind	PS-provider	P-DATA indication primitive
RT-TRind	RTSE	RT-TRANSFER indication primitive
RT-TRcnf+	RTSE	positive RT-TRANSFER confirm primitive
RT-TRcnf−	RTSE	negative RT-TRANSFER confirm primitive
RT-TPind	RTSE	RT-TURN-PLEASE indication primitive
RT-TGind	RTSE	RT-TURN-GIVE indication primitive
AA-REL	RTSE ACSE	RT-CLOSE response primitive or RT-CLOSE confirm primitive positive A-RELEASE response primitive or A-RELEASE confirm primitive
AA-ABreq	ROPM	abort application-association
AA-ABind	ROPM-TR	application-association aborted
ABORTind	RTSE ACSE	RT-P-ABORT indication primitive or the RT-U-ABORT indication primitive A-ABORT indication primitive or A-P-ABORT indication primitive

TABLE A-2/X.229

ROPM states

Abbreviated name	Name and description
STA01	idle; unassociated
STA02	associated

TABLE A-3/X.229

ROPM-TR states

Abbreviated name	Name and description
STA10	idle; unassociated
STA20	associated, token assigned, no transfer
STA21	associated, token assigned, transfer in progress
STA22	associated, token not assigned, no transfer
STA23	associated, token not assigned, transfer required
STA100	idle; unassociated
STA200	associated

TABLE A-4/X.229

Outgoing event list

Abbreviated name	Target	Name and description
RO-INVind	ROSE-user	RO-INVOKE indication primitive
RO-RESind	ROSE-user	RO-RESULT indication primitive
RO-ERRind	ROSE-user	RO-ERROR indication primitive
RO-RJUind	ROSE-user	RO-REJECT-U indication primitive
RO-RJPind	ROSE-user	RO-REJECT-P indication primitive
ROIV	ROPM-peer	RO-INVOKE APDU as user-data on a TRANSreq event
RORS	ROPM-peer	RO-RESULT APDU as user-data on a TRANSreq event
ROER	ROPM-peer	RO-ERROR APDU as user-data on a TRANSreq event
RORJu	ROPM-peer	RO-REJECT user-reject APDU as user-data on a TRANSreq event
RORJp	ROPM-peer	RO-REJECT provider-reject APDU as user-data on a TRANSreq event
TRANSreq	ROPM-TR	transfer request for an APDU
TRANSind	ROPM	transfer indication of an APDU
P-DATAreq	PS-provider	P-DATA request primitive
RT-TRreq	RTSE	RT-TRANSFER request primitive
RT-TPreq	RTSE	RT-TURN-PLEASE request primitive
RT-TGreq	RTSE	RT-TURN-GIVE request primitive
AA-ABreq	ROPM-TR	abort application-association
AA-ABind	ROPM	application-association aborted
ABORTreq	RTSE ACSE	RT-U-ABORT request primitive A-ABORT request primitive

TABLE A-5/X.229

Predicates

Code	Name and description
p1	unacceptable APDU is not RORJ APDU and number of rejects does not exceed locally specified value
p2	Token initially assigned to ROPM-TR

TABLE A-6/X.229

ROPM state table

	STA01	STA02
AA-ESTAB	STA02	
RO-INVreq		ROIV STA02
RO-RESreq		RORS STA02
RO-ERRreq		ROER STA02
RO-RJUreq		RORJu STA02
ROIV		RO-INVind STA02
RORS		RO-RESind STA02
ROER		RO-ERRind STA02
RORJu		RO-RJUind STA02
RORJp		RO-RJPind STA02
APDUua		p1: RORJp STA02 ¬ p1: AA-ABreq STA01
AA-ABind		STA01
AA-REL		STA01

TABLE A-7/X.229

ROPM-TR state table for transfer by RTSE

	STA10	STA20	STA21	STA22	STA23
AA-ESTAB	p2: STA20 ¬ p2: STA22				
TRANSreq		RT-TRreq STA21		RT-TPreq STA23	
RT-TRcnf+			STA20		
RT-TRcnf−			RO-RJPind STA20		
RT-TRind				TRANSind STA22	TRANSind STA23
RT-TPind		RT-TGreq STA22	STA21		
RT-TGind				STA20	RT-TRreq STA21
AA-ABreq		ABORTreq STA10	RO-RJPind ABORTreq STA10	ABORTreq STA10	RO-RJPind ABORTreq STA10
ABORTind		AA-ABind STA10	RO-RJPind AA-ABind STA10	AA-ABind STA10	RO-RJPind AA-ABind STA10
AA-REL		STA10	RO-RJPind STA10	STA10	RO-RJPind STA10

TABLE A-8/X.229

ROPM-TR state table for transfer by presentation service

	STA100	STA200
AA-ESTAB	STA200	
TRANSreq		P-DATAreq STA200
P-DATAind		TRANSind STA200
AA-ABreq		ABORTreq STA100
ABORTind		AA-ABind STA100
AA-REL		STA100

ANNEX B

(to Recommendation X.229)

Differences between this Recommendation and Recommendation X.410-1984

This Annex is not an integral Part of this Recommendation.

This Annex describes the technical differences between the notation and protocol for Remote Operations of this Recommendation and the corresponding notation and protocol of Recommendation X.410-1984.

B.1 *Macros*

B.1.1 *New Macros*

1) *Add*: BIND macro and UNBIND macro

B.1.2 *OPERATION Macro*

1) Value Notation

```
Change: From:  INTEGER
        To:    CHOICE
               { INTEGER,
                 OBJECT IDENTIFIER}
```

2) Named Type in Result production

Change: From: mandatory
To: optional

3) *Add*: Productions for linked Operations

B.1.3 *ERROR Macro*

1) Value Notation see B.1.2 item 1.

B.2 *Application protocol data units*

B.2.1 *APDUs*

1) Choice alternative

Change: From: explicit tagging
To: implicit tagging

B.2.2 *Invoke*

1) *Add*: optional linked-ID element to SEQUENCE

2) argument element

Change: From: mandatory
To: optional

B.2.3 *Return result*

1) *Add*: Field operation-value and SEQUENCE

2) result element

Change: From: mandatory
To: optional

B.2.4 *Reject*

1) Invoke Problem

Add: values (3) to (7) inclusive

B.3 *Procedures and mapping*

B.3.1 *Mapping onto used services*

1) *Add:* Mapping onto Presentation service if RTSE is absent in the Application Context

2) *Add:* Mapping for BIND and UNBIND

B.4 *Interworking between 84 and 88 implementation*

Due to B.2.1 item 1 and B.2.3 item 1 interworking between 84 and 88 implementations is not possible. However the first change was indicated in the X.400-Series Implementators Guide Version 5.

ANNEX C

(to Recommendation X.229)

Summary of assigned object identifier values

This Annex is not an integral Part of this Recommendation.

This Annex summarizes the object identifier values assigned in Recommendations X.219 and X.229.

```
{ joint-iso-ccitt remote-operations(4) notation (0) }
                          -- ASN.1 module defined in X.219

{ joint-iso-ccitt remote-operations(4) apdus (1) }
                          -- ASN.1 module defined in X.229

{ joint-iso-ccitt remote-operations(4) notation-extensions (2) }
                          -- ASN.1 module defined dans X.219

{ joint-iso-ccitt remote-operations(4) aseiD (3) }
                          -- ASE identifier defined in X.229

{ joint-iso-ccitt remote-operations(4) aseiD-ACSE (4) }
                          -- ASE identifier defined in X.219
```

PROCEDURE FOR THE EXCHANGE OF PROTOCOL IDENTIFICATION DURING VIRTUAL CALL ESTABLISHMENT ON PACKET SWITCHED PUBLIC DATA NETWORKS

(Malaga-Torremolinos, 1984)

The CCITT,

considering

(a) that Recommendation X.25 defines the 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 circuit;

(b) that Recommendation X.32 defines the interface between data terminal equipment (DTE) and data circuit-terminating equipment (DCE) for terminals operating in the packet mode and accessing a packet switched public data network through a public switched network;

(c) that Recommendation X.29 defines the procedures for the exchange of control information and user data between a packet assembly/disassembly facility (PAD) and a packet mode DTE or another PAD;

(d) that Recommendation X.224 defines the Transport Layer Protocol for Open Systems Interconnection for CCITT Applications, which includes provision of a Transport Protocol Identification; and

(e) that there is a need to identify and/or negotiate at virtual call establishment the protocol which is to operate above the packet level,

unanimously recommends

that for data terminal equipment operating in the packet mode the mechanism for identification and/or negotiation of the protocol which is to operate above the packet level should be as specified below.

1 Introduction

This Recommendation defines the procedures for the exchange of protocol identification during virtual call establishment on packet switched public data networks. This procedure operates above the packet level of Recommendation X.25. It makes use of bits 8 and 7 of the first octet of the user data field in the call set-up packets defined in Recommendation X.25.

2 Call user data field

The following procedure applies to the call user data field of Call Request and Incoming Call packets.

If the call user data field is present, the use and format of this field are determined by bits 8 and 7 of the first octet of this field (see Note below).

If bits 8 and 7 of the first octet of the call user data field are 00, a portion of the call user data field is used for protocol identification in accordance with other Recommendations such as Recommendation X.29 and Recommendation X.224.

If bits 8 and 7 of the first octet of the call user data field are 01, a portion of the call user data field may be used for protocol identification in accordance with specifications of Administrations.

If bits 8 and 7 of the first octet of the call user data field are 10, a portion of the call user data field may be used for protocol identification in accordance with specifications of international user bodies.

If bits 8 and 7 of the first octet of the call user data field are 11, no constraints are placed on the use by the DTE of the remainder of the call user data field.

Users are cautioned that if bits 8 and 7 of the first octet of the call user data field have any value other than 11, a protocol may be identified that is implemented within public data networks.

Note – When a virtual call is being established between two packet mode DTEs, the network does not act on any part of the call user data field.

3 Called user data field

The following procedure applies to the called user data field of Call Accepted and Call Connected packets used in conjunction with the fast select facility.

If the called user data field is present, the use and format of this field is determined by bits 8 and 7 of the first octet of this field (see Note below).

If bits 8 and 7 of the first octet of the called user data field are 00, a portion of the called user data field is used for protocol identification in accordance with other Recommendations.

If bits 8 and 7 of the first octet of the called user data field are 01, a portion of the called user data field may be used for protocol identification in accordance with specifications of Administrations.

If bits 8 and 7 of the first octet of the called user data field are 10, a portion of the called user data field may be used for protocol identification in accordance with specifications of international user bodies.

If bits 8 and 7 of the first octet of the called user data field are 11, no constraints are placed on the use by the DTE of the remainder of the called user data field.

Users are cautioned that if bits 8 and 7 of the first octet of the called user data field have any value other than 11, a protocol may be identified that is implemented within public data networks.

Note – When a virtual call is being established between two packet mode DTEs, the network does not act on any part of the called user data field.

SECTION 4

CONFORMANCE TESTING

Recommendation X.290

OSI CONFORMANCE TESTING METHODOLOGY AND FRAMEWORK FOR PROTOCOL RECOMMENDATIONS FOR CCITT APPLICATIONS[1]

(Melbourne, 1988)

The CCITT,

considering

(a) that Recommendation X.200 defines the Reference Model of Open Systems for CCITT Applications;

(b) that the objective of OSI will not be completely achieved until systems dedicated to CCITT applications can be tested to determine whether they conform to the relevant OSI protocol Recommendations;

(c) that standardized test suites should be developed for each OSI protocol Recommendation as a means to:

- obtain wide acceptance and confidence in conformance test results produced by different testers,
- provide confidence in the interoperability of equipments which passed the standardized conformance tests;

(d) the need for defining an international Recommendation to specify the framework and general principles for the specification of conformance test suites and the testing of protocol implementations,

unanimously recommends

(1) that the general principles, definition of terms and concepts of OSI protocol conformance testing shall be in accordance with Part 1 of this Recommendation;

(2) that the test methods, test suites and test notation shall be in accordance with Part 2 of this Recommendation.

[1] Recommendation X.290 was developed in close collaboration with the ISO/IEC effort on OSI Conformance Testing Methodology and Framework. At the time of publication, Recommendation X.290 was aligned with the texts of DP 9646/1 and DP 9646/2. Since this work was at an early stage of development, changes should be expected. Consequently, users should exercise caution in applying this Recommendation.

CONTENTS

Part 1 – General concepts

SECTION 1 – *Terminology*

SECTION 2 – *Overview*

Part 2 – Abstract test suite specification

SECTION 1 – *Requirements on protocol specifiers*

SECTION 2 – *Requirements on abstract test suite specifiers*

Part 1 – General concepts

0 Introduction

The objective of OSI will not be completely achieved until systems can be tested to determine whether they conform to the relevant "OSI or related CCITT X-series or T-series" (hereafter abbreviated to "OSI*") protocol "standard(s) or Recommendation(s)" (hereafter abbreviated to "Recommendation(s)*").

Standardized test suites should be developed for each OSI* protocol Recommendation, for use by suppliers or implementors in self-testing, by users of OSI products, by the Administrations* or other third party testers. This should lead to comparability and wide acceptance of test results produced by different testers, and thereby minimize the need for repeated conformance testing of the same system.

The standardization of test suites requires international definition and acceptance of a common testing methodology and appropriate testing methods and procedures. It is the purpose of this Recommendation to define the methodology, to provide a framework for specifying conformance test suites and define the procedures to be followed during testing.

Conformance testing involves testing both the capabilities and behaviour of an implementation and checking what is observed against the conformance requirements in the relevant Recommendation(s)* and against what the implementor states the implementation's capabilities are.

Conformance testing does not include assessment of the performance nor the robustness or reliability of an implementation. It cannot give judgements on the physical realization of the abstract service primitives, how a system is implemented, how it provides any requested service, nor the environment of the protocol implementation. It cannot, except in an indirect way, prove anything about the logical design of the protocol itself.

The purpose of conformance testing is to increase the probability that different implementations are able to interwork. This is achieved by verifying them by means of a protocol test suite, thereby increasing the confidence that each implementation conforms to the protocol specification. Confidence in conformance to a protocol specification is particularly important when equipment supplied by different vendors is required to interwork.

However, it should be borne in mind that the complexity of most protocols makes exhaustive testing impractical on both technical and economic grounds. Also, testing cannot guarantee conformance to a specification since it detects errors rather than their absence. Thus conformance to a test suite alone cannot guarantee interworking. What it does do is give confidence that an implementation has the required capabilities and that its behaviour conforms consistently in representative instances of communication.

It should be noted that the OSI reference model for CCITT applications (Recommendation X.200) states (in § 4.3):

> "Only the external behaviour of Open Systems is retained as the standard of behaviour of real Open Systems".

This means that although aspects of both internal and external behaviour are described in OSI* Recommendations*, only the requirements on external behaviour have to be met by real open systems. Although some of the methods defined in this Recommendation do impose certain constraints on the implementor, for example that there be some means of realizing control and observation at one or more service access points, it should be noted that other methods defined herein do not impose such constraints.

However, in the case of partial OSI* end-systems which provide OSI* protocols up to a specific layer boundary, it is desirable to test both the external behaviour of the implemented protocol entities and the potential of those entities for supporting correct external behaviour in higher layers.

Detailed investigation of relative benefits, efficiency and constraints of all methods is addressed in various parts of this Recommendation. However, any organization contemplating the use of test methods defined in this Recommendation in a context such as certification should carefully consider the constraints on applicability and the benefits of the different possible test methods.

Testing is voluntary as far as ISO/CCITT is concerned. Requirements for testing in procurement and other external contracts are not a matter for standardization.

1 Scope and field of application

1.1 This Recommendation specifies a general methodology for testing the conformance to OSI* protocol Recommendation(s)* of products in which the Recommendation(s)* are claimed to be implemented. The methodology also applies to testing conformance to transfer syntax Recommendation(s)* to the extent that can be determined by testing each in combination with a specific OSI* protocol.

1.2 This Recommendation is structured into two separate parts:

Part 1 identifies the different phases of conformance testing process, these phases being characterized by four major roles. These roles are:

a) the specification of abstract test suites for particular OSI* protocols;

b) the derivation of executable test suites and associated testing tools;

c) the role of a client of a test laboratory, having an implementation of OSI* protocols to be tested;

d) the operation of conformance testing, culminating in the production of a conformance test report which gives the results in terms of the Recommendation(s)* and the test suite(s) used.

Additionally, this Part provides tutorial material, together with definition of concepts and terms.

Part 2 defines the requirements and guidance for the specification of abstract test suites for OSI* protocols.

1.3 In both Parts of this Recommendation, the scope is limited to include only such information as is necessary to meet the following objectives:

a) to achieve an adequate level of confidence in the tests as a guide to conformance;

b) to achieve comparability between the results of the corresponding tests applied in different places at different times;

c) to facilitate communication between the parties responsible for the roles described above.

1.4 One such aspect of this scope involves the framework for development of OSI* test suites. For example:

a) how they should relate to the various types of conformance requirement;

b) the types of test to be standardized and the types not needing standardization;

c) the criteria for selecting tests for inclusion in a conformance test suite;

d) the notation to be used for defining tests;

e) the structure of a test suite.

1.5 Certification, an administrative procedure which may follow conformance testing, is outside the scope of this Recommendation. Requirements for procurement and contracts are also outside the scope of this Recommendation.

1.6 The Physical layer and Media Access Control protocols are outside the field of application of this Recommendation.

2 References

Recommendation X.200 – *Reference model of open systems interconnection for CCITT applications* (see also ISO 7498).

Recommendation X.210 – *Open systems interconnection layer service definition conventions* (see also ISO TR 8509).

Recommendation X.209 – *Specification of basic encoding rules for abstract syntax notation one (ASN.1)* (see also ISO 8825).

3 Definitions

3.1 *Reference model definitions*

This Recommendation is based upon the concepts developed in reference model of open systems interconnection for CCITT applications (CCITT X.200), and makes use of the following terms defined in that Recommendation:

a) (N)-entity

b) (N)-service

c) (N)-layer

d) (N)-protocol

e) (N)-service-access-point

f) (N)-relay

g) (N)-protocol-data-unit

h) (N)-protocol-control-information

i) (N)-user-data

j) real open system

k) subnetwork

l) application-entity

m) application-service-element

n) transfer syntax

o) physical layer

p) data link layer

q) network layer

r) transport layer

s) session layer

t) presentation layer

u) application layer

v) systems-management

w) application-management

x) layer-management

3.2 *Terms defined in other Recommendations*

This Recommendation uses the following terms defined in the OSI Service Conventions (Recommendation X.210):

a) service-user

b) service-provider

This Recommendation uses the following term defined in the ASN.1 – Basic Encoding Rules Recommendation (Recommendation X.209):

c) encoding

3.3 *Conformance testing definitions*

For the purposes of this Recommendation the definitions in §§ 3.4 to 3.8 apply.

3.4 *Basic terms*

3.4.1 **implementation under test (IUT)**

That part of a real open system which is to be studied by testing, which should be an implementation of one or more OSI* protocols in an adjacent user/provider relationship.

3.4.2 **system under test (SUT)**

The real open system in which the IUT resides.

3.4.3 **dynamic conformance requirements**

All those requirements (and options) which determine what observable behaviour is permitted by the relevant OSI* Recommendation(s)* in instances of communication.

3.4.4 **static conformance requirements**

Constraints which are placed in OSI* Recommendations* to facilitate interworking by defining the requirements for the capabilities of an implementation.

Note – Static conformance requirements may be at a broad level, such as the grouping of functional units and options into protocol classes, or at a detailed level, such as the ranges of values that are to be supported for specific parameters or timers.

3.4.5 **capabilities of an IUT**

That set of functions and options in the relevant protocol(s) and, if appropriate, that set of facilities and options of the relevant service definition which are supported by the IUT.

3.4.6 **protocol implementation conformance statement (PICS)**

A statement made by the supplier of an OSI* implementation or system, stating the capabilities and options which have been implemented, and any features which have been omitted.

3.4.7 **PICS proforma**

A document, in the form of a questionnaire, designed by the protocol specifier or conformance test suite specifier, which when completed for an OSI* implementation or system becomes the PICS.

3.4.8 **protocol implementation extra information for testing (PIXIT)**

A statement made by a supplier or implementor of an IUT which contains or references all of the information (in addition to that given in the PICS) related to the IUT and its testing environment, which will enable the test laboratory to run the appropriate test suite against the IUT.

3.4.9 **PIXIT proforma**

A document, in the form of a questionnaire, provided by the test laboratory, which when completed during the preparation for testing becomes a PIXIT.

3.4.10 **conforming implementation**

An IUT which is shown to satisfy both static and dynamic conformance requirements, consistent with the capabilities stated in the PICS.

3.4.11 **system conformance statement**

A document summarizing which OSI* Recommendations* are implemented and to which conformance is claimed.

3.4.12 **client**

The organization that submits a system or implementation for conformance testing.

3.4.13 **test laboratory**

An organization that carries out conformance testing. This can be a third party, a user organization, an Administration*, or an identifiable part of the supplier organization.

3.5 *Types of testing*

3.5.1 **active testing**

The application of a test suite to an SUT, under controlled conditions, with the intention of observing the consequent actions of the IUT.

3.5.2 **passive testing**

The observation of PDU activity on a link, and checking whether or not the observed behaviour is allowed by the relevant Recommendation(s)*.

3.5.3 **multi-layer testing**

Testing the behaviour of a multi-layer IUT as a whole, rather than testing it layer by layer.

3.5.4 **embedded testing**

Testing the behaviour of a single layer within a multi-layer IUT without accessing the layer boundaries for that layer within the IUT.

3.5.5 **basic interconnection testing**

Limited testing of an IUT to determine whether or not there is sufficient conformance to the main features of the relevant protocol(s) for interconnection to be possible, without trying to perform thorough testing.

3.5.6 **capability testing**

Testing to determine the capabilities of an IUT.

Note – This involves checking all mandatory capabilities and those optional ones that are stated in the PICS as being supported, but not checking those optional ones which are stated in the PICS as not supported by the IUT.

3.5.7 **static conformance review**

A review of the extent to which the static conformance requirements are met by the IUT, by comparing the static conformance requirements expressed in the relevant Recommendation(s)* with the PICS and the results of any associated capability testing.

3.5.8 **behaviour testing**

Testing the extent to which the dynamic conformance requirements are met by the IUT.

3.5.9 **conformance testing**

Testing the extent to which an IUT is a conforming implementation.

3.5.10 **conformance assessment process**

The complete process of accomplishing all conformance testing activities necessary to enable the conformance of an implementation or a system to one or more OSI* Recommendations* to be assessed. It includes the production of the PICS and PIXIT documents, preparation of the real tester and the SUT, the execution of one or more test suites, the analysis of the results and the production of the appropriate system and protocol conformance test reports.

3.6 *Terminology of test suites*

3.6.1 **abstract test method**

The description of how an IUT is to be tested, given at an appropriate level of abstraction to make the description independent of any particular implementation of testing tools, but with enough detail to enable tests to be specified for this method.

3.6.2 **abstract testing methodology**

An approach to describing and categorizing abstract test methods.

3.6.3 **abstract test case**

A complete and independent specification of the actions required to achieve a specific test purpose, defined at the level of abstraction of a particular abstract test method. It includes a preamble and a postamble to ensure starting and ending in a stable state (i.e., a state which can be maintained almost indefinitely, such as the "idle" state or "data transfer" state) and involves one or more consecutive or concurrent connections.

Note 1 – The specification should be complete in the sense that it is sufficient to enable a verdict to be assigned unambiguously to each potentially observable outcome (i.e., sequence of test events).

Note 2 – The specification should be independent in the sense that it should be possible to execute the derived executable test case in isolation from other such test cases (i.e., the specification should always include the possibility of starting and finishing in the "idle" state – that is without any existing connections except permanent ones). For some test cases, there may be pre-requisites in the sense that execution might require some specific capabilities of the IUT, which should have been confirmed by results of the test cases executed earlier.

3.6.4 **executable test case**

A realization of an abstract test case.

Note – In general the use of the word "test" will imply its normal English meaning. Sometimes it may be used as an abbreviation for abstract test case or executable test case. The context should make the meaning clear.

3.6.5 **test purpose**

A description of the objective which an abstract test case is designed to achieve.

3.6.6 **generic test case**

A specification of the actions required to achieve a specific test purpose, defined by a test body together with a description of the initial state in which the test body is to start.

3.6.7 **preamble**

The test steps needed to define the path from the starting stable state of the test case up to the initial state from which the test body will start.

3.6.8 **test body**

The set of test steps that are essential in order to achieve the test purpose and assign verdicts to the possible outcomes.

3.6.9 **postamble**

The test steps needed to define the paths from the end of the test body up to the finishing stable state for the test case.

3.6.10 **test step**

A named subdivision of a test case, constructed from test events and/or other test steps, and used to modularize abstract test cases.

3.6.11 **test event**

An indivisible unit of test specification at the level of abstraction of the specification (e.g., sending or receiving a single PDU).

3.6.12 **test suite**

A complete set of test cases, possibly combined into nested test groups, that is necessary to perform conformance testing or basic interconnection testing for an IUT or protocol within an IUT.

3.6.13 **test case**

A generic, abstract or executable test case.

3.6.14 **test group**

A named set of related test cases.

3.6.15 **generic test suite**

A test suite composed of generic test cases, with the same coverage as the complete set of test purposes for the particular protocol, this being the set or a superset of the test purposes of any particular abstract test suite for the same protocol.

3.6.16 **abstract test suite**

A test suite composed of abstract test cases.

3.6.17 **executable test suite**

A test suite composed of executable test cases.

3.6.18 **conformance test suite**

A test suite for conformance testing of one or more OSI* protocols.

Note – It should cover both capability testing and behaviour testing. It may be qualified by the adjectives: abstract, generic or executable, as appropriate. Unless stated otherwise, an "abstract test suite" is meant.

3.6.19 **basic interconnection test suite**

A test suite for basic interconnection testing of one or more OSI* protocols.

3.6.20 **selected abstract test suite**

The subset of an abstract test suite selected using a specific PICS.

3.6.21 **selected executable test suite**

The subset of an executable test suite selected using a specific PICS and corresponding to a selected abstract test suite.

3.6.22 **parameterized abstract test case**

An abstract test case in which all appropriate parameters have been supplied with values in accordance with a specific PICS and PIXIT.

3.6.23 **parameterized executable test case**

An executable test case in which all appropriate parameters have been supplied with values in accordance with a specific PICS and PIXIT.

3.6.24 **parameterized abstract test suite**

A selected abstract test suite in which all test cases have been made parameterized abstract test cases for the appropriate PICS and PIXIT.

3.6.25 **parameterized executable test suite**

A selected executable test suite in which all test cases have been made parameterized executable test cases for the appropriate PICS and PIXIT, and corresponding to a parameterized abstract test suite.

3.7 *Terminology of results*

3.7.1 **repeatability (of results)**

Characteristic of a test case, such that repeated executions on the same IUT lead to the same verdict, and by extension a characteristic of a test suite.

3.7.2 **comparability (of results)**

Characteristic of conformance assessment processes, such that their execution on the same IUT, in different test environments, leads to the same overall summary.

3.7.3 **outcome**

A sequence of test events together with the associated input/output, either identified by an abstract test case specifier, or observed during test execution.

3.7.4 **foreseen outcome**

An outcome identified or categorized in the abstract test case specification.

3.7.5 **unforeseen outcome**

An outcome not identified or categorized in the abstract test case specification.

3.7.6 **verdict**

Statement of "pass", "fail" or "inconclusive" concerning conformance of an IUT with respect to a test case that has been executed and which is specified in the abstract test suite.

3.7.7 **system conformance test report (SCTR)**

A document written at the end of the conformance assessment process, giving the overall summary of the conformance of the system to the set of protocols for which conformance testing was carried out.

3.7.8 **protocol conformance test report (PCTR)**

A document written at the end of the conformance assessment process, giving the details of the testing carried out for a particular protocol, including the identification of the abstract test cases for which corresponding executable test cases were run and for each test case the test purpose and verdict.

3.7.9 **valid test event**

A test event which is allowed by the protocol Recommendation*, being both syntactically correct and occurring or arriving in an allowed context in an observed outcome.

3.7.10 **syntactically invalid test event**

A test event which syntactically is not allowed by the protocol Recommendation*.

Note – The use of "invalid test event" is deprecated.

3.7.11 **inopportune test event**

A test event which, although syntactically correct, occurs or arrives at a point in an observed outcome when not allowed to do so by the protocol Recommendation*.

3.7.12 **"pass" verdict**

A verdict given when the observed outcome satisfies the test purpose and is valid with respect to the relevant Recommendation(s)* and with respect to the PICS.

3.7.13 **"fail" verdict**

A verdict given when the observed outcome is syntactically invalid or inopportune with respect to the relevant Recommendation(s)* or the PICS.

3.7.14 **"inconclusive" verdict**

A verdict given when the observed outcome is valid with respect to the relevant Recommendation(s)* but prevents the test purpose from being accomplished.

3.7.15 **conformance log**

A record of sufficient information necessary to verify verdict assignments as a result of conformance testing.

3.8 *Terminology of test methods*

3.8.1 **point of control and observation (PCO)**

A point at which control and observation is specified in a test case.

3.8.2 **lower tester**

The abstraction of the means of providing, during test execution, control and observation at the appropriate PCO either below the IUT or remote from the IUT, as defined by the chosen abstract test method.

3.8.3 **upper tester**

The abstraction of the means of providing, during test execution, control and observation of the upper service boundary of the IUT, plus the control and observation of any relevant abstract local primitive.

3.8.4 **abstract (N)-service-primitive ((N)-ASP)**

An implementation independent description of an interaction between a service-user and a service-provider at an (N)-service boundary, as defined in an OSI* service definition Recommendation*.

3.8.5 **abstract local primitive (ALP)**

An abbreviation for a description of control and/or observation to be performed by the upper tester, which cannot be described in terms of ASPs but which relates to events or states defined within the protocol Recommendation(s)* relevant to the IUT.

Note – The PIXIT will indicate whether or not a particular ALP can be realized within the SUT. The ability of the SUT to support particular ALPs as specified in the PIXIT will be used as a criterion in the test selection process.

3.8.6 **test coordination procedures**

The rules for cooperation between the lower and upper testers during testing.

3.8.7 **test management protocol**

A protocol which is used as a realization of the test coordination procedures for a particular test suite.

3.8.8 **local test methods**

Abstract test methods in which the PCOs are directly at the layer boundaries of the IUT.

3.8.9 external test methods

Abstract test methods in which the lower tester is separate from the SUT and communicates with it via an appropriate lower layer service-provider.

Note – The service-provider is immediately beneath the (lowest layer) protocol which is the focus of the testing, and may involve multiple OSI layers.

3.8.10 distributed test method

An external test method in which there is a PCO at the layer boundary at the top of the IUT.

3.8.11 coordinated test method

An external test method for which a standardized test management protocol is defined as the realization of the test coordination procedures, enabling the control and observation to be specified solely in terms of the lower tester activity, including the control and observation of test management PDUs.

3.8.12 remote test method

An external method in which there is neither a PCO above the IUT nor a standardized test management protocol; some requirements for test coordination procedures may be implied or informally expressed in the abstract test suite but no assumption is made regarding their feasibility or realization.

3.8.13 real tester

The realization of the lower tester, plus either the definition or the realization of the upper tester, plus the definition of the test coordination procedures, as appropriate to a particular test method.

3.8.14 test realizer

An organization which takes responsibility for providing, in a form independent of client and IUT, the means of testing IUTs in conformance with the abstract test suite.

4 Abbreviations

For the purposes of this Recommendation the following abbreviations apply:

Administration*	Administration or recognized private operating agency
ALP	abstract local primitive
ASP	abstract service primitive
DTE	data terminal equipment
IUT	implementation under test
OSI	open systems interconnection
OSI*	OSI or related CCITT X-series or T-series Recommendations
PCO	point of control and observation
PCTR	protocol conformance test report
PDU	protocol data unit
PICS	protocol implementation conformance statement
PIXIT	protocol implementation extra information for testing
BBSAP	service access point
SCTR	system conformance test report
Recommendation*	Standard or Recommendation
SUT	system under test
TM-PDU	test management PDU

SECTION 2 – *Overview*

5 The meaning of conformance in OSI

5.1 *Introduction*

In the context of OSI*, a real system is said to exhibit conformance if it complies with the requirements of applicable OSI* Recommendations* in its communication with other real systems.

Applicable OSI* Recommendations* include protocol Recommendations*, and transfer syntax Recommendations* inasmuch as they are implemented in conjunction with protocols.

OSI* Recommendations* form a set of interrelated Recommendations* which together define behaviour of open systems in their communication. Conformance of a real system will, therefore, be expressed at two levels, conformance to each individual Recommendation*, and conformance to the set.

Note – If the implementation is based on a predefined set of Recommendations*, often referred to as a functional standard or profile, the concept of conformance can be extended to specific requirements expressed in the functional standard or profile, as long as they do not conflict with the requirements of the base Recommendations*.

5.2 *Conformance requirements*

5.2.1 The conformance requirements in a Recommendation* can be:

a) mandatory requirements: these are to be observed in all cases;

b) conditional requirements: these are to be observed if the conditions set out in the Recommendation* apply;

c) options: these can be selected to suit the implementation, provided that any requirements applicable to the option are observed. More information on options is provided in Annex A.

For example, CCITT essential facilities are mandatory requirements; additional facilities can be either conditional or optional requirements.

Note – The CCITT terms "essential facilities" and "additional facilities" need to be considered in the context of the scope of the CCITT Recommendation concerned; in many cases, essential facilities are mandatory for networks but not for DTEs.

5.2.2 Furthermore, conformance requirements in a Recommendation* can be stated

a) positively: they state what shall be done;

b) negatively (prohibitions): they state what shall not be done.

5.2.3 Finally, conformance requirements fall into two groups:

a) static conformance requirements;

b) dynamic conformance requirements;

these are discussed in §§ 5.3. and 5.5, respectively.

5.3 *Static conformance requirements*

Static conformance requirements are those that define the allowed minimum capabilities of an implementation, in order to facilitate interworking. These requirements may be at a broad level, such as the grouping of functional units and options into protocol classes, or at a detailed level, such as a range of values that have to be supported for specific parameters of timers.

Static conformance requirements and options in OSI* Recommendations* can be of two varieties:

a) those which determine the capabilities to be included in the implementation of the particular protocol;

b) those which determine multi-layer dependencies, e.g., those which place constraints on the capabilities of the underlying layers of the system in which the protocol implementation resides. These are likely to be found in upper layer Recommendations*.

All capabilities not explicitly stated as static conformance requirements are to be regarded as optional.

5.4 *Protocol implementation conformance statement (PICS)*

To evaluate the conformance of a particular implementation, it is necessary to have a statement of the capabilities and options which have been implemented, and any features which have been omitted, so that the implementation can be tested for conformance against relevant requirements, and against those requirements only. Such a statement is called a Protocol Implementation Conformance Statement (PICS).

In a PICS there should be a distinction between the following categories of information which it may contain:

a) information related to the mandatory, optional and conditional static conformance requirements of the protocol itself;

b) information related to the mandatory, optional and conditional static conformance requirements for multi-layer dependencies.

If a set of interrelated OSI* protocol Recommendations* has been implemented in a system, a PICS is needed for each protocol. A System Conformance Statement will also be necessary, summarizing all protocols in the system for each of which a distinct PICS is provided.

5.5 *Dynamic conformance requirements*

Dynamic conformance requirements are all those requirements (and options) which determine what observable behaviour is permitted by the relevant OSI* Recommendation(s)* in instances of communication. They form the bulk of each OSI* protocol Recommendation*. They define the set of allowable behaviours of an implementation or real system. This set defines the maximum capability that a conforming implementation or real system can have within the terms of the OSI* protocol Recommendation*.

A system exhibits dynamic conformance in an instance of communication if its behaviour is a member of the set of all behaviours permitted by the relevant OSI* protocol Recommendation(s)* in a way which is consistent with the PICS.

5.6 *A conforming system*

A conforming system or implementation is one which is shown to satisfy both static and dynamic conformance requirements, consistent with the capabilities stated in the PICS, for each protocol declared in the System Conformance Statement.

5.7 *Interworking and conformance*

5.7.1 The primary purpose of conformance testing is to increase the probability that different implementations are able to interwork.

Successful interworking of two or more real open systems is more likely to be achieved if they all conform to the same subset of an OSI* Recommendation*, or to the same selection of OSI* Recommendations*, than if they do not.

In order to prepare two or more systems to interwork successfully, it is recommended that a comparison be made of the System Conformance Statements and PICSs of these systems.

If there is more than one version of a relevant OSI* Recommendation* indicated in the PICSs, the differences between the versions need to be identified and their implications for consideration, including their use in combination with other Recommendations*.

5.7.2 While conformance is a necessary condition, it is not on its own a sufficient condition to guarantee interworking capability. Even if two implementations conform to the same OSI* protocol Recommendation*, they may fail to interwork because of factors outside the scope of that Recommendation.

Trial interworking is recommended in order to detect these factors. Further information to assist interworking between two systems can be obtained by extending the PICS comparison to other relevant information, including test reports and PIXIT (see § 6.2). The comparison can focus on:

a) additional mechanisms claimed to work around known ambiguities or deficiencies not yet corrected in the Recommendations* or in peer real systems, e.g. solution of multi-layer problems;

b) selection of free options which are not taken into account in the static conformance requirements of the Recommendations*;

c) the existence of timers not specified in the Recommendation* and their associated values.

Note – The comparison can be made between two individual systems, between two or more types of product, or, for the PICS comparison only, between two or more specifications for procurement, permissions to connect, etc.

6 Conformance and testing

6.1 *Objectives of conformance testing*

6.1.1 *Introduction*

Conformance testing as discussed in this Recommendation is focused on testing for conformance to OSI* protocol Recommendations*. However, it also applies to testing for conformance to OSI* transfer syntax Recommendations*, to the extent that this can be carried out by testing the transfer syntax in combination with an OSI* protocol.

In principle, the objective of conformance testing is to establish whether the implementation being tested conforms to the specification in the relevant Recommendation*. Practical limitations make it impossible to be exhaustive, and economic considerations may restrict testing still further.

Therefore, this Recommendation distinguishes four types of testing, according to the extent to which they provide an indication of conformance:

a) basic interconnection tests, which provide *prima facie* evidence that an IUT conforms;

b) capability tests, which check that the observable capabilities of the IUT are in accordance with the static conformance requirements and the capabilities claimed in the PICS;

c) behaviour tests, which endeavour to provide testing which is as comprehensive as possible over the full range of dynamic conformance requirements specified by the Recommendation*, within the capabilities of the IUT;

d) conformance resolution tests, which probe in depth the conformance of an IUT to particular requirements, to provide a definite yes/no answer and diagnostic information in relation to specific conformance issues; such tests are not standardized.

6.1.2 *Basic interconnection tests*

6.1.2.1 Basic interconnection tests provide limited testing of an IUT in relation to the main features in a Recommendation*, to establish that there is sufficient conformance for interconnection to be possible, without trying to perform thorough testing.

6.1.2.2 Basic interconnection tests are appropriate:

a) for detecting severe cases of non-conformance;

b) as a preliminary filter before undertaking more costly tests;

c) to give a *prima facie* indication that an implementation which has passed full conformance tests in one environment still conforms in a new environment (e.g. before testing an (N)-implementation, to check that a tested (N – 1)-implementation has not undergone any severe change due to being linked to the (N)-implementation);

d) for use by users of implementations, to determine whether the implementations appear to be usable for communication with other conforming implementations, e.g. as a preliminary to data interchange.

6.1.2.3 Basic interconnection tests are inappropriate:

a) as a basis for claims of conformance by the supplier of an implementation;

b) as a means of arbitration to determine causes for communications failure.

6.1.2.4 Basic interconnection tests should be standardized as either a very small test suite or a subset of a conformance test suite (including capability and behaviour tests). They can be used on their own or together with a conformance test suite. The existence and execution of basic interconnection tests are optional.

6.1.3 *Capability tests*

6.1.3.1 Capability tests provide limited testing of each of the static conformance requirements in a Recommendation*, to ascertain what capabilities of the IUT can be observed and to check that those observable capabilities are valid with respect to the static conformance requirements and the PICS.

6.1.3.2 Capability tests are appropriate:

a) to check as far as possible the consistency of the PICS with the IUT;

b) as a preliminary filter before undertaking more in-depth and costly testing;

c) to check that the capabilities of the IUT are consistent with the static conformance requirements;

d) to enable efficient selection of behaviour tests to be made for a particular IUT;

e) when taken together with behaviour tests, as a basis for claims of conformance.

6.1.3.3 Capability tests are inappropriate:

a) on their own, as a basis for claims of conformance by the supplier of an implementation;

b) for testing in detail the behaviour associated with each capability which has been implemented or not implemented;

c) for resolution of problems experienced during live usage or where other tests indicate possible non-conformance even though the capability tests have been satisfied.

6.1.3.4 Capability tests are standardized within a conformance test suite. They can either be separated into their own test group(s) or merged with the behaviour tests.

6.1.4 *Behaviour tests*

6.1.4.1 Behaviour tests test an implementation as thoroughly as is practical, over the full range of dynamic conformance requirements specified in a Recommendation*. Since the number of possible combinations of events and timing of events is infinite, such testing cannot be exhaustive. There is a further limitation, namely that these tests are designed to be run collectively in a single test environment, so that any faults which are difficult or impossible to detect in that environment are likely to be missed. Therefore, it is possible that a non-conforming implementation passes the conformance test suite; one aim of the test suite design is to minimize the number of times that this occurs.

6.1.4.2 Behaviour tests are appropriate, when taken together with capability tests, as a basis for the conformance assessment process.

6.1.4.3 Behaviour tests are inappropriate for resolution of problems experienced during live usage or where other tests indicate possible non-conformance even though the behaviour tests have been satisfied.

6.1.4.4 Behaviour tests are standardized as the bulk of a conformance test suite.

Note – Behaviour tests include tests for valid behaviour by the IUT in response to valid, inopportune and syntactically invalid protocol behaviour by the real tester. This includes testing the rejection by the IUT of attempts to use features (capabilities) which are stated in the PICS as being not implemented. Thus, capability tests do not need to include tests for capabilities omitted from the PICS.

6.1.5 *Conformance resolution tests*

6.1.5.1 Conformance resolution tests provide diagnostic answers, as near to definitive as possible, to the resolution of whether an implementation satisfies particular requirements. Because of the problems of exhaustiveness noted in § 6.1.4.1, the definite answers are gained at the expense of confining tests to a narrow field.

6.1.5.2 The test architecture and test method will normally be chosen specifically for the requirements to be tested, and need not be ones that are generally useful for other requirements. They may even be ones that are regarded as being unacceptable for (standardized) abstract conformance test suites, e.g. involving implementation-specific methods using, say, the diagnostic and debugging facilities of the specific operating system.

6.1.5.3 The distinction between behaviour tests and conformance resolution tests may be illustrated by the case of an event such as a Reset. The behaviour tests may include only a representative selection of conditions under which a Reset might occur, and may fail to detect incorrect behaviour in other circumstances. The conformance resolution tests would be confined to conditions under which incorrect behaviour was already suspected to occur, and would confirm whether or not the suspicions were correct.

6.1.5.4 Conformance resolution tests are appropriate:

a) for providing a yes/no answer in a strictly confined and previously identified situation (e.g. during implementation development, to check whether a particular feature has been correctly implemented, or during operational use, to investigate the cause of problems);

b) as a means for identifying and offering resolutions for deficiencies in a current conformance test suite.

6.1.5.5 Conformance resolution tests are inappropriate as a basis for judging whether or not an implementation conforms overall.

6.1.5.6 Conformance resolution tests are not standardized.

Note on § 6.1 – As a by-product of conformance testing, errors and deficiencies in protocol Recommendations* may be identified.

6.2 *Protocol implementation extra information for testing (PIXIT)*

In order to test a protocol implementation, the test laboratory will require information relating to the IUT and its testing environment in addition to that provided by the PICS. This "Protocol Implementation eXtra Information for Testing" (PIXIT) will be provided by the client submitting the implementation for testing, as a result of consultation with the test laboratory.

The PIXIT may contain the following information:

a) information needed by the test laboratory in order to be able to run the appropriate test suite on the specific system (e.g., information related to the test method to be used to run the test cases, addressing information);

b) information already mentioned in the PICS and which needs to be made precise (e.g. a timer value range which is declared as a parameter in the PICS should be specified in the PIXIT);

c) information to help determine which capabilities stated in the PICS as being supported are testable and which are untestable;

d) other administrative matters (e.g. the IUT identifier, reference to the related PICS).

The PIXIT should not conflict with the appropriate PICS.

The abstract test suite specifier, test realizer and test laboratory will all contribute to the development of the PIXIT proforma.

6.3 *Conformance assessment process outline*

6.3.1 The main feature of the conformance assessment process is a configuration of equipment allowing exchanges of information between the IUT and a real tester. These are controlled and observed by the real tester.

6.3.2 In conceptual outline, conformance testing should include several steps, involving both static conformance reviews and live testing phases, culminating in the production of a test report which is as thorough as is practical.

6.3.3 These steps are:

a) analysis of the PICS;

b) test selection and parameterization;

c) basic interconnection testing (optional);

d) capability testing;

e) behaviour testing;

f) review and analysis of test results;

g) synthesis, conclusions and conformance test report production.

These are illustrated in Figure 1/X.290 Part 1.

Prior to the execution of any of the tests, the IUT's PICS and PIXIT are input to the test case selection and parameterization process.

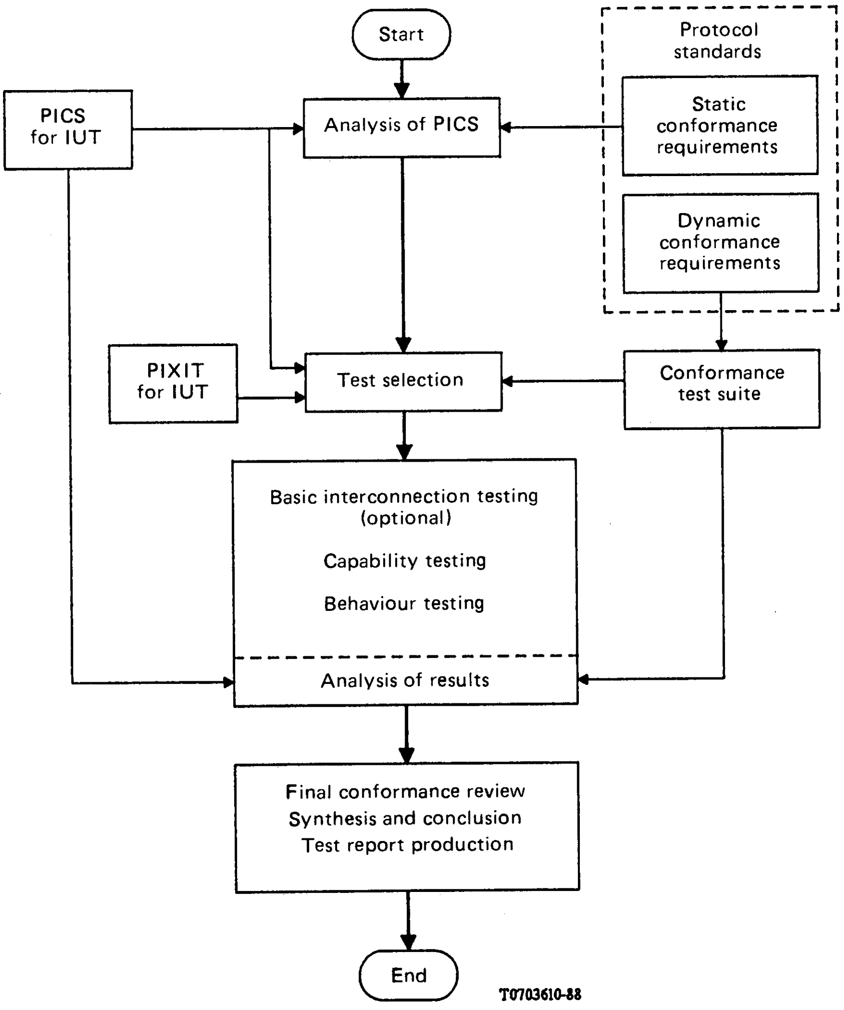


FIGURE 1/X.290, Part 1

Conformance assessment process outline

6.4 *Analysis of results*

6.4.1 *General*

6.4.1.1 *Outcomes and verdicts*

The observed outcome (of the test execution) is the series of events which occurred during execution of a test case; it includes all input to and output from the IUT at the points of control and observation.

The foreseen outcomes are identified and defined by the abstract test case specification taken in conjunction with the protocol Recommendation*. For each test case, there may be one or more foreseen outcomes. Foreseen outcomes are defined primarily in abstract terms.

A verdict is a statement of pass, fail or inconclusive to be associated with every foreseen outcome in the abstract test suite specification.

The analysis of results is performed by comparing the observed outcomes with foreseen outcomes.

The verdict assigned to an observed outcome is that associated with the matching foreseen outcome. If the observed outcome is unforeseen then the abstract test suite specification will state what default verdict shall be assigned.

The means by which the comparison of the observed outcomes with the foreseen outcomes is made is outside the scope of this Recommendation.

Note – Amongst the possibilities are:

a) manual or automated comparison (or a mixture);

b) comparison at or after execution time;

c) translating the observed outcomes into abstract terms for comparison with the foreseen outcomes or translating the foreseen outcomes into the terms used to record the observed outcomes.

The verdict will be pass, fail or inconclusive:

a) pass means that the observed outcome satisfies the test purpose and is valid with respect to the relevant Recommendation(s)* and with respect to the PICS;

b) fail means that the observed outcome is syntactically invalid or inopportune with respect to the relevant Recommendation(s)* or the PICS;

c) inconclusive means that the observed outcome is valid with respect to the relevant Recommendation(s)* but prevents the test purpose from being accomplished.

The verdict assigned to a particular outcome will depend on the test purpose and the validity of the observed protocol behaviour.

The verdicts made in respect of individual test cases will be synthesized into an overall summary for the IUT based on the test cases executed.

6.4.1.2 *Conformance test reports*

The results of conformance testing will be documented in a set of conformance test reports. These reports will be of two types: a System Conformance Test Report (SCTR), and a Protocol Conformance Test Report (PCTR).

The SCTR, which will always be provided, gives an overall summary of the conformance status of the SUT, with respect to its single or multi-layer IUT. A standard proforma for the SCTR is for further study.

The PCTR, one of which will be issued for each protocol tested in the SUT, documents all of the results of the test cases giving references to the conformance logs which contain the observed outcomes. The PCTR also gives reference to all necessary documents relating to the conduct of the conformance assessment process for that protocol.

A standard proforma for the PCTR is for further study. The ordered list of test cases to be used in the PCTR will be specified in the conformance test suite Recommendation*.

6.4.2 *Repeatability of results*

In order to achieve the objective of credible conformance testing, it is clear that the result of executing a test case on an IUT should be the same whenever it is performed. Statistically, it may not be possible to perform a complete conformance test suite and observe outcomes which are completely identical to those obtained on another occasion: unforeseen events do occur, and this is a feature of the environments involved. Nevertheless, at the test case level, it is very important that every effort is made by the test specifiers and test laboratories to minimize the possibility that a test case produces different outcomes on different occasions.

6.4.3 *Comparability of results*

In order to achieve the ultimate objectives of conformance testing, the overall summary concerning conformance of an IUT has to be independent of the test environment in which the testing takes place. That is to say, the standardization of all of the procedures concerned with conformance testing should result in a comparable overall summary being accorded to the IUT, whether the testing is done by the supplier, a user, or by any third party test house. There are a large number of factors to be studied to achieve this, of which some of the more important are:

a) careful design of the abstract test case specification to give flexibility where appropriate, but show which requirements have to be met (which is the subject of this Recommendation);

b) careful specification of the real tester which should be used to run the test suite; again this specification should give flexibility where appropriate, but show which requirements have to be met, including all test coordination procedures (if any);

c) careful specification of the procedure to be followed in determining how the contents of the PICS are to be used in the analysis of outcomes of test cases; there should be no room for "optimistic" interpretation;

d) careful specification of the procedures to be followed by test laboratories as regards the repetition of a test case before making a final verdict for that test purpose;

e) a proforma for a conformance test report;

f) careful specification of the procedures necessary when synthesizing an overall summary.

6.4.4 *Auditability of results*

For legal reasons, as well as others, it may be necessary to review the observed outcomes from the execution of a conformance test suite in order to make sure that all procedures have been correctly followed. Whether or not analysis has been carried out in a manual or automatic mode, it is essential that all inputs, outputs, and other test events are careful logged, and the analysis of the results recorded. In some cases this may be the responsibility of the test realizer, who may elect to include the test criteria in the conformance log, as well as all outcomes. In others, it may be the responsibility of the test laboratory, which might be required to follow all standard procedures concerning the recording of results.

Note – As far as auditability is concerned, some automatic procedures would be preferred, but in the event it should be appreciated that from a legal standpoint such automatic procedures would have to be accredited themselves, if they are to be credible.

7 **Test methods**

7.1 *Introduction*

The testing of a given OSI* protocol can require the use of several test methods, as systems under test can come in several configurations, and vary in terms of their ability to allow ways of producing effects applicable to a layer boundary.

This section first characterizes the features of the system under test which are to be taken into consideration, next defines the possible test methods in abstract terms, and finally provides guidance on their applicability to real systems.

7.2 *Classification of real open systems and IUTs for conformance testing*

7.2.1 *Classification of systems under test*

7.2.1.1 There is a relation between the test methods and the configurations of the real open systems to be tested. The appropriate test methods vary according to:

a) the main function of the system (end-system or relay-system);

b) which layers use OSI* protocols;

c) whether the alternative of non-OSI* protocols is also available.

7.2.1.2 The following configurations of systems have been identified for the purposes of conformance testing, as illustrated in Figures 2/X.290 to 4/X.290, Part 1. Configurations 1 to 3 are the basic configurations of systems under test (SUTs):

a) Configuration 1: 7-layer open system (end-system)

These systems use OSI* Recommendation* protocols in all 7 layers.

b) Configuration 2: Partial (N)-open system (end-system)

These systems use OSI* Recommendation* protocols in layers 1 to N.

c) Configuration 3: Open relay-systems

These use OSI* protocols in layers 1 to 3 (Network relay-systems) or 1 to 7 (Application relay-systems).

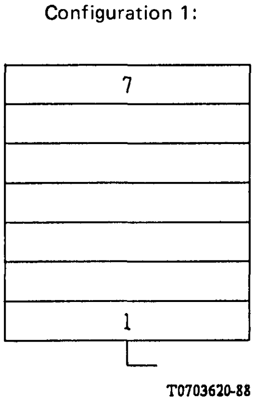


FIGURE 2/X.290, Part 1

7-layer open system

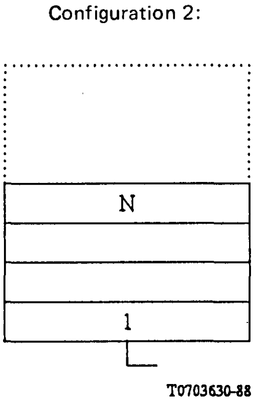


FIGURE 3/X.290, Part 1

Partial (N)-open system

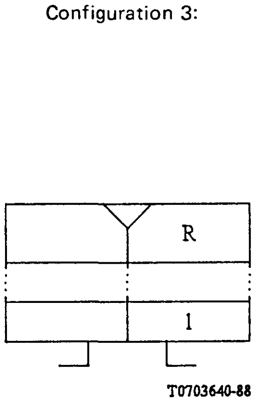


FIGURE 4/X.290, Part 1

Open-relay system

7.2.1.3 Other configurations can be derived from the basic configurations.

A SUT can be a combination of basic configurations 1 and 2, allowing the alternative of using OSI* and non-OSI* protocols above layer N (see Figure 5/X.290, Part 1).

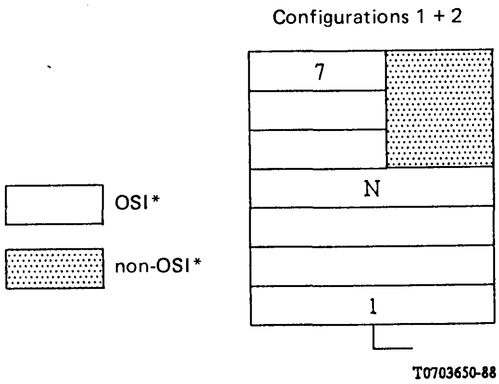


FIGURE 5/X.290, Part 1

Combination of 7-layer and partial (N)-open systems

7.2.2 *Identification of the implementation under test (IUT)*

An implementation under test (IUT) is that part of a real open system which is to be studied by conformance testing. It should be an implementation of one or more adjacent OSI* protocols.

IUTs can be defined for configurations 1 and 2 of SUTs as single-layer IUTs (one single layer of the SUT is to be tested), or as multi-layer IUTs (a set of any number of adjacent layers of the SUT to be tested in combination).

An IUT defined in an open relay-system will include at least the layer which provides the relay function.

When OSI* and non-OSI* protocols exist in a system, the IUT(s) will be defined for the OSI* mode(s) of operation. Testing non-OSI* protocols is outside the scope of this Recommendation.

Clients and test laboratories will agree on what part of the SUT will be considered to be the IUT.

7.3 *Abstract testing methodology – General*

Test methods need to refer to an abstract testing methodology, based upon the OSI reference model. Considering first end-systems (7-layer or partial (N)-open systems) and single layer IUTs within these systems, abstract test methods are described in terms of what outputs from the IUT are observed and what inputs to it can be controlled. More specifically, an abstract test method is described by identifying the points closest to the IUT at which control and observation are to be exercized.

The OSI* protocol Recommendations* define allowed behaviour of a protocol entity (i.e. the dynamic conformance requirements) in terms of the protocol data units (PDUs) and the abstract service primitives (ASPs) both above and below that entity. Thus the behaviour of an (N)-ASPs and (N – 1)-ASPs (the latter including the (N)-PDUs).

If an IUT comprises more than one protocol entity, the required behaviour can be defined in terms of the ASPs above and below the IUT, including the PDUs of the protocols in the IUT.

The starting point for developing test methods is the conceptual testing architecture, illustrated in Figure 6/X.290, Part 1. It is a "black-box" active testing architecture, based on the definition of behaviour required of the IUT.

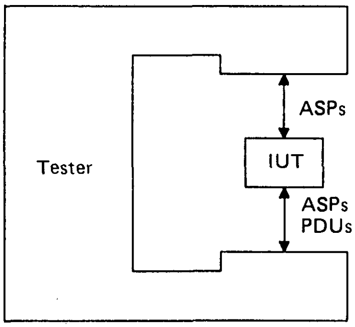
The action of the tester shown in Figure 6/X.290, Part 1, can either be applied locally, in which case there is a direct coupling within the system under test, or externally via a link or network. The two sets of interactions, above and below the IUT, can, in practice, be observed and controlled from several different points, locally or externally.

The possible points of control and observation (PCOs) are identified by three factors:

a) whether it is the ASPs or PDUs which are observed and controlled;

b) the layer identity of the ASPs or PDUs concerned;

c) whether they are controlled and observed within the system under test or in a system remote from the system under test – if the latter then the ASPs are distinguished by the addition of a double-quote character (").

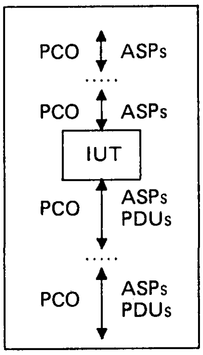


T0703660-88

FIGURE 6/X.290, Part 1

Conceptual testing architecture

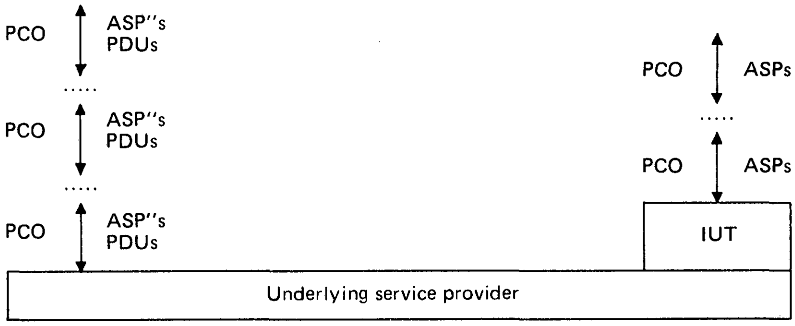
Possible PCOs within the SUT are illustrated in Figure 7a)/X.290, Part 1. Possible PCOs, when the activity below the IUT is controlled and observed externally are illustrated in Figure 7b)/X.290, Part 1. It can be seen from these figures that there is a multiplicity of possible PCOs in different layers, which offer different degrees of control and observation of IUT behaviour. This Recommendation makes a selection from this set of possible PCOs, defining a limited number of abstract test methods.



T0703670-88

FIGURE 7 a)/X.290, Part 1

Possible PCOs within an SUT



T0703680-88

FIGURE 7 b)/X.290, Part 1

Possible PCOs for external testing

If control and observation below, or external from the IUT is specified in terms of ASPs, it will include control and observation of the PDUs carried by those ASPs; but if it is specified in terms of PDUs (at layer N) then the ASPs (at layer N − 1) are not considered to be controlled or observed.

It is assumed that the ASP activity below the IUT can at least be observable and controllable via the peer activity in a remote testing system – i.e. the corresponding ASP"s. Thus when the ASPs below the IUT are not controllable nor observable locally, conformance testing can be carried out externally, provided that the underlying service offered is sufficiently reliable for control and observation to take place remotely.

It is possible that the ASP activity above the IUT might not be controllable nor observable, in which case this activity is said to be hidden.

SUTs are not required to provide access to layer boundaries. However, the possible provision of such access and the possible positions of such boundaries with respect to the layers of the IUT are factors to be taken into consideration in the definition of the test methods, which may take advantage of this access to define the test suites in terms of the corresponding ASPs. It does not matter whether the accessible boundaries are accessed via service access points (SAPs) or via some other PCOs.

Figure 8/X.290, Part 1 shows examples of IUTs, with respect to layer boundary accessibility.

Note – In addition, a conformance test suite Recommendation* may define "abstract local primitives". These are used to specify control and observation of events or states which are referred to in the protocol Recommendation* but which are internal to the IUT and which cannot be expressed in terms of ASPs. They are abbreviations for text descriptions of control and observation to be performed by the upper tester.

Similar consideration apply to relay systems (see Part 2 of the Recommendation for details).

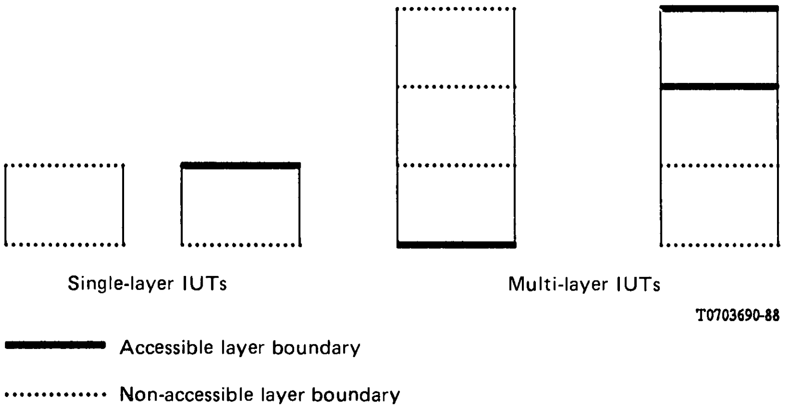


FIGURE 8/X.290, Part 1

Examples of IUT configurations

7.4 *Abstract testing functions*

The definition of abstract test methods requires that the PCOs be distributed over two abstract testing functions, the lower and upper testers.

The lower tester is the abstraction of the means of providing, during test execution, control and observation at the appropriate PCO either below the IUT or remote from the IUT, as defined by the chosen abstract test method. Thus, it is the testing function related to the control and observation of the lower boundary of the IUT. If the action of the tester is local to the SUT, the lower tester will take the place of the lower part of the SUT. If the action of the tester is external to the SUT, the lower tester will rely on the (N – 1)-Service, provided jointly by the lower tester itself, a link and the SUT.

The upper tester is the abstraction of the means of providing, during test execution, control and observation of the upper service boundary of the IUT and of any relevant abstract local primitive.

There is a need for cooperation between the upper tester and the lower tester; the rules for such cooperation are called the test coordination procedures.

The test methods will vary in the way that they specify the test coordination procedures. In some cases it is possible to define a test management protocol to provide the coordination between the upper and lower testers. In other cases it is only possible to describe the requirements to be met by the test coordination procedures without specifying what mechanisms might be used to realize them.

7.5 *Overview of abstract test methods*

7.5.1 *End-system IUTs*

For the IUTs defined within end-system SUTs (configurations 1 and 2 in Figures 2/X.290, Part 1 and 3/X.290, Part 1) four categories of abstract test methods are defined, one local, and three external ones which assume the lower tester is located remotely from the SUT and connected to it by a link or network.

7.5.2 *The local test methods*

The local abstract test methods define the PCOs as being at the service boundaries above and below the IUT. The test events are specified in terms of the ASPs above the IUT and the ASPs and PDUs below the IUT, as illustrated in Figure 9a)/X.290, Part 1. Abstractly, a lower tester is considered to observe and control the ASPs and PDUs below the IUT, while an upper tester observes and controls the ASPs above the IUT. Requirements to be met by test coordination procedures used to coordinate the realizations of the upper and lower testers are defined in the abstract conformance test suites, although the test coordination procedures themselves are not.

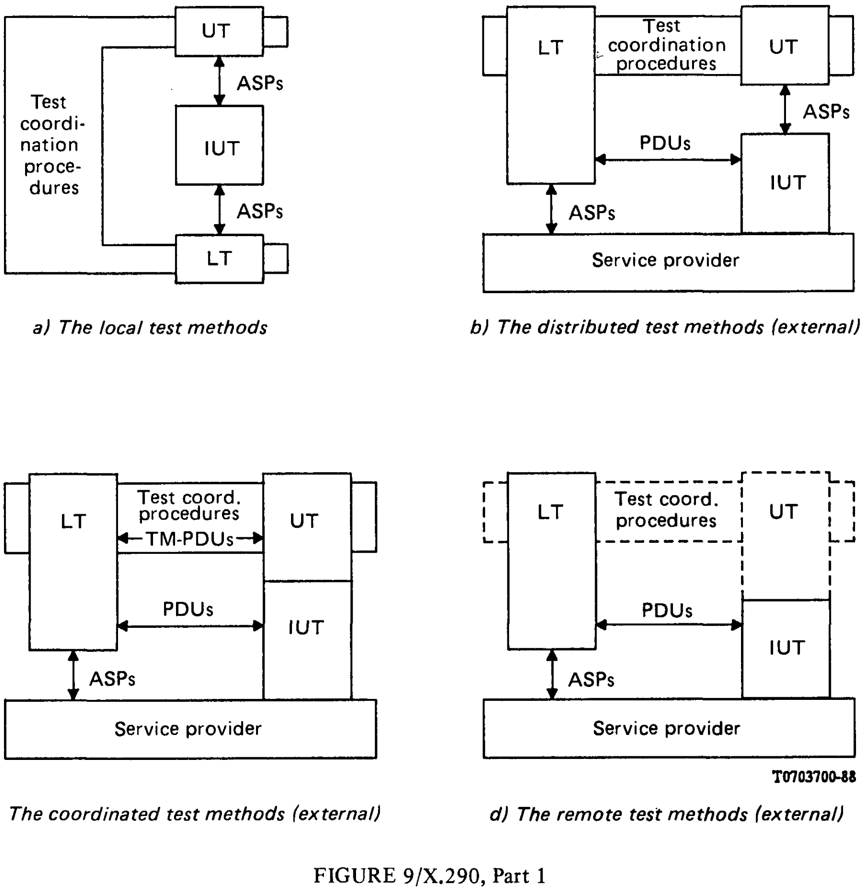
7.5.3 *External test methods*

The external test methods use control and observation of the ASPs below the IUT by means of a lower tester separated from the SUT, together with control and observation of the ASPs above the IUT. Three different categories of external abstract test methods are defined, which are referred to as distributed, coordinated, and remote. They vary according to the level of requirement or standardization put on the test coordination procedures, on the access to the layer boundary above the IUT, and on the requirements on an upper tester. They are illustrated in Figures 9b), c) and d)/X.290, Part 1.

The coordinated test method requires that the test coordination procedures used to coordinate the realization of the upper and lower testers be achieved by means of test management protocols. The other two methods do not make any assumptions about the realization of the test coordination procedures.

The distributed and coordinated test methods require specific functions from the upper tester above the IUT. The remote method does not.

The distributed method requires access to the upper boundary of the IUT. The other two methods do not.



a) The local test methods

b) The distributed test methods (external)

c) The coordinated test methods (external)

d) The remote test methods (external)

FIGURE 9/X.290, Part 1

Overview of abstract test methods

7.5.4 *Variants of end-system test methods*

Each category of test methods has variants which can be applied to single-layer IUTs or to multi-layer IUTs. For multi-layer IUTs in which the protocols are to be tested layer by layer, embedded variants can be used.

All abstract test methods for end-systems are fully specified in section 8 of Part 2 of this Recommendation, including their single-layer, multi-layer and embedded variants where applicable.

7.5.5 *Relay-system IUTs*

For open relay-systems, two test methods are defined, loop-back and transverse. These are fully specified in section 8 of Part 2 of this Recommendation.

7.6 *Applicability of test methods to real open systems*

The architecture and stage of development of a real open system determines the applicability of test methods to it.

Local test methods are useful for systems under development, when their architecture permits the isolation of an IUT, be it single-layer or multi-layer.

External test methods are useful for testing complete or partial end-systems which can attach to telecommunications networks.

Coordinated test methods apply where it is possible to implement a standardized test management protocol in an upper tester in the SUT, above the IUT.

Remote test methods apply when it is possible to make use of some functions of the SUT to control the IUT during testing, instead of using a specific upper tester.

Distributed test methods apply when it is necessary to allow complete freedom for the implementation of the test coordination procedures between the SUT and the lower tester, but to specify in detail the control and observation requirements at both ends.

Single-layer test methods are the most appropriate methods for testing the majority of the protocol conformance requirements.

Multi-layer test methods will be used when conformance to true multi-layer dynamic conformance requirements is to be tested.

Embedded test methods permit the application of single-layer testing to all layers of a multi-layer IUT.

For 7-layer open systems, the preferred methods are the incremental use of appropriate external single-layer embedded methods with the following PCOs:

a) the upper interface of the application layer as provided by the 7-layer open system, when applicable;

b) successively, each SAP (or corresponding PCO if there is no SAP as such) below the protocol which is the focus of the testing, as controlled and observed in the external lower tester, starting from the lowest protocol of the IUT and working upwards.

7.7 *Applicability of the test methods to OSI* protocols and layers*

The test methods defined in this Recommendation apply to all layers, with the exception of the Physical layer and the Media Access Control protocols which are outside the scope of this Recommendation. Appendix I of this Recommendation provides guidance on the applicability of test methods to all other layers.

8 **Test suites**

8.1 *Structure*

Test suites have a hierarchical structure (see Figure 10/X.290, Part 1) in which an important level is the test case. Each test case has a narrowly defined purpose, such as that of verifying that the IUT has a certain required capability (e.g. the ability to support certain packet sizes) or exhibit a certain required behaviour (e.g. behave as required when a particular event occurs in a particular state).

Within a test suite, nested test groups are used to provide a logical ordering of the test cases. Test groups may be nested to an arbitrary depth. They may be used to aid planning, development, understanding or execution of the test suite.

Test cases are modularized by using named subdivisions called test steps. Each test case comprises at least one test step: the ordering of events covered by the test purpose ("test body"). It may include further test steps to put the IUT into the state required for the test body to start from (the "preamble") or return to a quiescent state after the test body has finished (the "postamble").

For practical reasons, common test steps may be grouped together into test step libraries. Test step libraries may be structured into nested sets of test steps, to any depth of nesting. Test step libraries may be associated with the whole test suite or with a particular test group or test case.

Furthermore, all test steps consist of an ordering of other test steps and/or test events (e.g. the transfer of a single PDU or ASP to or from the IUT). All test steps are, therefore, equivalent to an ordering of test events (after expansion of the inner test steps).

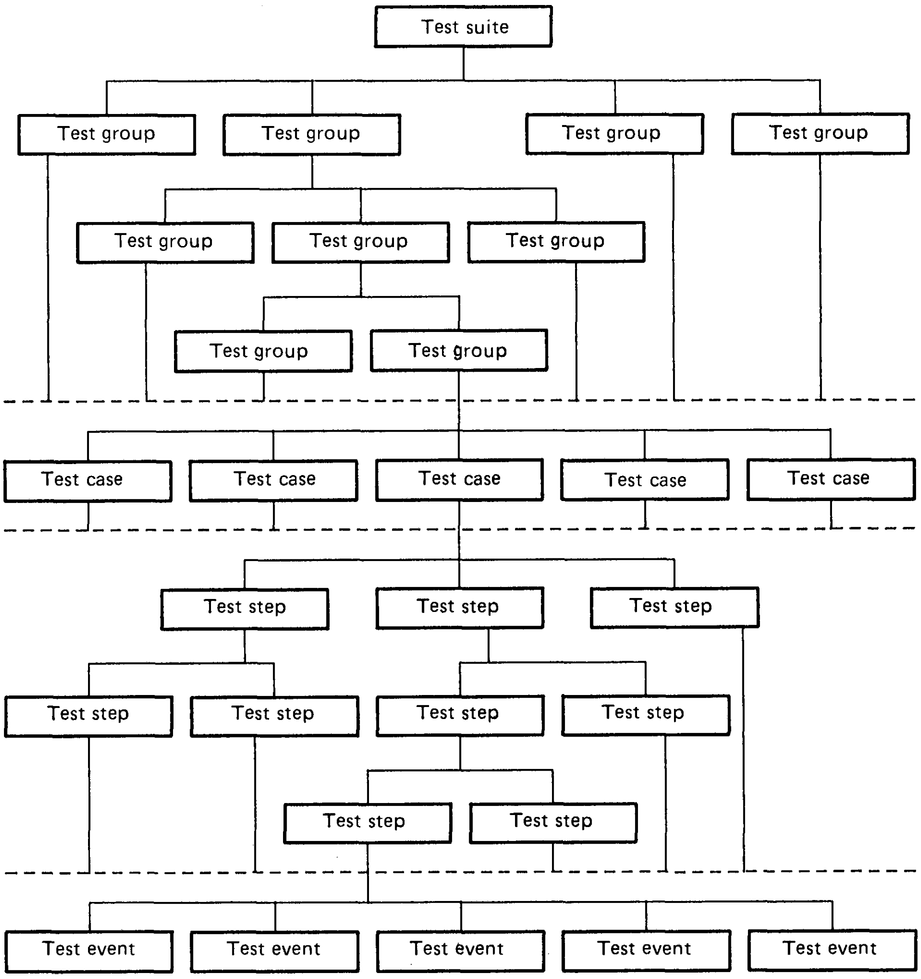


FIGURE 10/X.290, Part 1

Test suite structure

8.2 *Generic, abstract and executable test cases*

8.2.1 A generic test case is one which:

a) provides a refinement of the test purpose;

b) specifies that all sequences of test events (paths) in the test body which correspond to verdicts of "pass", "fail" and "inconclusive", using a specialized notation;

c) is used as the common root of corresponding abstract test cases for different abstract test suites for the same protocol;

d) includes a description of the initial state in which the test body should start, in lieu of a preamble;

e) need not describe the postamble;

f) is specified using the style of either the Remote or Distributed Single-layer test methods.

8.2.2 An abstract test case is derived from a generic case and the relevant protocol specification; it:

a) specifies the test case in terms of a particular test method;

b) adds a more precise specification for sequences of events which are only described informally in the generic test case;

c) adds the sequences of test events required to achieve the objectives of the preamble and postamble of the generic test case using a specialized notation.

8.2.3 An executable test case is derived from an abstract test case, and is in a form which allows it to be run on a real tester for testing a real implementation.

8.2.4 The terms generic, abstract and executable are used to describe test suites, which comprise generic, abstract and executable test cases, respectively.

8.2.5 A generic test suite has the coverage of the set or a superset of all possible abstract test suites for a particular protocol.

9 Relationships between concepts and roles

Figure 11/X.290, Part 1 is a pictorial representation of the relation between the various Recommendations and the processes of producing generic, abstract and executable test suites and test reports.

Part 2 concerns the production of testable protocol Recommendations and abstract test suite Recommendations. Part 1 provides general concepts and definitions.

Note – Other aspects of the conformance assessment process, such as executable test derivation, preparation of the IUT, PICS and PIXIT by the client and test laboratory role are for further study.

10 Compliance

In this Recommendation, "compliance" refers to meeting the requirements specified by the Recommendation. This word is used as an attempt to eliminate confusion between *compliance* to this Recommendation and *conformance* of a protocol implementation to protocol Recommendations.

This part of the Recommendation contains no compliance requirements.

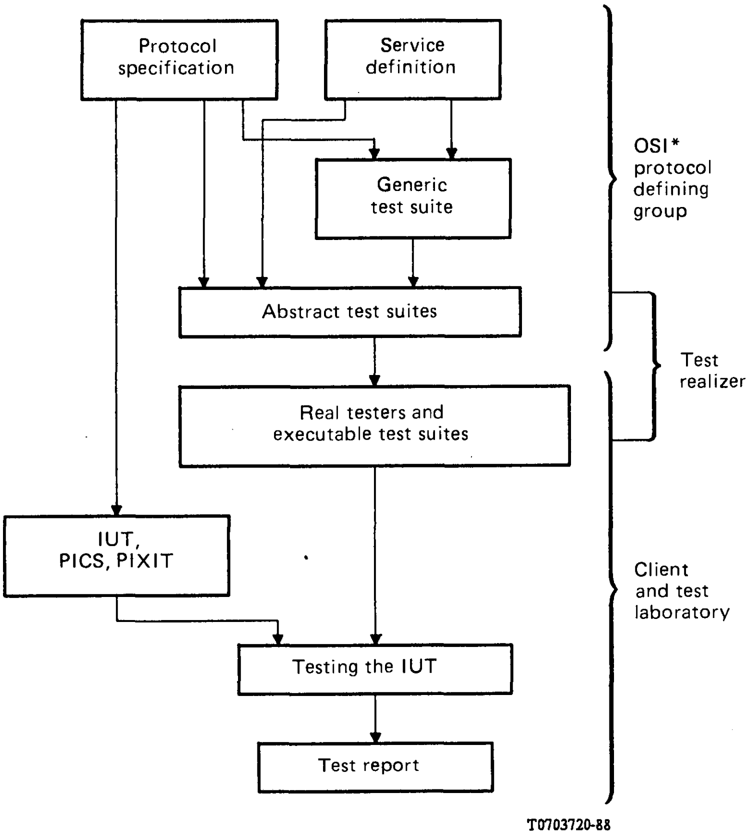


FIGURE 11/X.290, Part 1

Relationships between concepts and roles

Part 2 – Abstract test suite specification

0 Introduction

This part of the Recommendation provides a common approach to the specification of "OSI or related CCITT X-series or T-series" (hereafter abbreviated to "OSI*") conformance test suites at a level which is independent of the means of executing those test suites (hereafter called "abstract test suites"). This level of abstraction is suitable for standardization and facilitates the comparison of results produced by different organizations which run the corresponding executable test suites.

Section One recalls that there are requirements put on the OSI* protocol specifiers which have to be fulfilled before there can be an objective basis for the process of developing an abstract test suite. The need for consistent conformance sections and for PICS and PIXIT proformas in OSI* protocol Recommendations* is expressed.

Section Two describes the process of developing an abstract test suite, including the design criteria to be used and guidance on its structure and coverage. The possible abstract test methods are defined and guidance is given to help the test suite specifier to decide which method(s) to use in the production of a particular test suite. The relationship between abstract test suites for different methods is provided by a generic test suite which is independent of the test method. A test notation is defined and requirements and guidance are given on its use for specifying both generic and abstract test cases. These include the subdivision of test cases into test steps and the assignment of verdicts to outcomes.

The test suite specifier is also required to provide information to the test realizers, particularly on the constraints governing test case selection and ordering.

Finally, guidance and requirements are given on test suite maintenance.

1 Scope and field of application

This part of the Recommendation specifies the requirements and provides guidance for the production of system-independent conformance test suites for one or more OSI* Recommendations*. In particular, it applies to the production of all conformance test suite Recommendations* for OSI* protocols.

It applies to the production of conformance test cases which check the adherence of an implementation to the relevant static and/or dynamic conformance requirements by controlling and observing protocol behaviour. The abstract test methods included in this Recommendation are in fact capable of being used to specify any test case which can be expressed abstractly in terms of control and observation of protocol data units, abstract service primitives and abstract local primitives. Nevertheless, for some protocols, test cases may be needed which cannot be expressed in these terms. The specification of such test cases is outside the scope of this Recommendation, although those test cases may need to be included in a Recommendation* for a conformance test suite.

Note – For example, some static conformance requirements related to an application service may require testing techniques which are specific to that particular application.

The production of conformance test suites for multi-peer or physical layer protocols is outside the scope of this Recommendation.

The relation between abstract test specification and formal description techniques is outside the scope of this Recommendation.

2 References

Recommendation X.200 *Reference model of open systems interconnection for CCITT applications* (see also ISO 7498).

Recommendation X.214 *Transport service definition for open systems interconnection for CCITT applications* (see also ISO 8072).

Recommendation X.224 *Transport protocol specification for open systems interconnection for CCITT applications* (see also ISO 8073).

Recommendation X.210	*Open systems interconnection layer service definition conventions* (see also ISO TR 8509).
Recommendation X.208	*Specification of abstract syntax notation one (ASN.1)* (see also ISO 8824).
Recommendation X.209	*Specification of basic encoding rules for abstract syntax notation one (ASN.1)* (see also ISO 8825).
Recommendation X.290/1	*OSI conformance testing methodology and framework for protocol Recommendations for CCITT applications – Part 1: General concepts.*
ISO 8571-4	*Information processing systems – open systems interconnection – File protocol specification.*

3 Definitions

For the purposes of this Part of the Recommendation, all the definitions in Part 1 of the Recommendation apply.

4 Abbreviations

For the purposes of this Recommendation the abbreviations given in section 4 of Part 1 of this Recommendation apply. The following abbreviations also apply to this Recommendation:

ASP”	the abstract service primitive on the side of the service provider remote from the IUT
CM	coordinated multi-layer (test method)
CS	coordinated single-layer (test method)
CSE	coordinated single-layer embedded (test method)
DM	distributed multi-layer (test method)
DS	distributed single-layer (test method)
DSE	distributed single-layer embedded (test method)
FDT	formal description technique
LM	local multi-layer (test method)
LS	local single-layer (test method)
LSE	local single-layer embedded (test method)
RM	remote multi-layer (test method)
RS	remote single-layer (test method)
RSE	remote single-layer embedded (test method)
TTCN	tree and tabular combined notation
YL	loop-back (test method)
YT	transverse (test method)

5 Compliance

5.1 A protocol Recommendation* which complies with this Part of this Recommendation shall satisfy all the requirements stated in Section 1.

Note – Such compliance is a precondition for the protocol Recommendation* to be an effective basis for conformance testing of implementations.

5.2 An abstract test suite specification which complies with this part of this Recommendation:

a) shall be a conformance test suite;

b) shall be specified in a test notation standardized by ISO or CCITT;

c) shall satisfy all the requirements stated in Section 2.

5.3 It is recommended that the test notation used be the Tree and Tabular Combined Notation (TTCN). If TTCN is used, the abstract test suite shall comply with all the requirements in Annex D.

SECTION 1 – *Requirements on protocol specifiers*

6 Conformance requirements in OSI* Recommendations*

6.1 *Introduction*

The meaning of conformance in OSI* is discussed in Part 1 of this Recommendation. It is necessary that there be an unambiguous and objective understanding of the conformance requirements of an OSI* protocol or transfer syntax Recommendation*, as a prerequisite to the production of an abstract test suite for that Recommendation*. This section states the requirements on protocol specifiers to ensure that there is such an understanding of the conformance requirements.

6.2 *General requirements*

6.2.1 A clear distinction shall be made between static and dynamic conformance requirements. To avoid ambiguity, they should be stated separately from one another.

6.2.2 It shall be clear what conformance to the Recommendation* means, in the sense of what shall be done, what is permitted but not mandatory, and what shall not be implemented, to conform to the Recommendation*.

6.2.3 It shall always be decidable whether an instance of communication conforms dynamically or not.

For example, one should be able to look at a record of PDU activity and decide whether it is valid.

6.2.4 Requirements on the need to produce and content of a PICS shall be stated separately from the requirements on the protocol implementation itself.

6.3 *Conformance sections*

6.3.1 Each OSI* protocol and transfer syntax Recommendation* shall include a conformance section, which shall be expressed clearly and unambiguously.

6.3.2 Conformance sections shall distinguish between the following categories of information that they may contain:

a) references to sections which state dynamic conformance requirements;

b) static conformance requirements concerning the protocol implementation;

c) static conformance requirements concerning multi-layer dependencies;

d) what has to be stated in the PICS concerning (b) above;

e) what has to be stated in the PICS concerning (c) above;

f) what other information shall be provided (e.g. to assist testing) and whether this should be in the PICS or elsewhere.

6.3.3 In connection-oriented protocol Recommendations*, the conformance section should also include:

a) the option to support either the initiation of a connection or the acceptance of a connection, or both;

b) the requirement to be able to accept all correct sequences of PDUs received from peers, and respond with correct PDU sequences appropriate to the defined state of the connection;

c) the requirement to be able to respond correctly to all incorrect sequences of PDUs received, the response being appropriate to the defined state of the connection.

6.4 *Additional guidance for new protocol Recommendations**

It is recognized that although existing protocol Recommendations* can be improved by the progression of an addendum to add a PICS proforma and make the conformance section align with the requirements stated above, it is unrealistic to expect any greater improvement. However, new protocol Recommendations* should be developed using the additional guidance given in Annex B to make the task of understanding the conformance requirements easier and less prone to ambiguity.

7 PICS proformas

7.1 *Requirements on PICS proformas*

7.1.1 The specific requirements to be met by suppliers in respect of each PICS they provide shall normally be stated in the relevant protocol Recommendation*. The specification of these requirements shall include a PICS proforma. In exceptional circumstances, the PICS proforma may be found in the abstract test suite Recommendation* rather than in protocol Recommendation*; in particular, this applies when the PICS proforma has to cover different versions of the same protocol coming from both ISO and CCITT. In normal circumstances, the PICS proforma should be found in an annex to the protocol Recommendation* and referenced in the conformance section, and, if necessary, progressed as an addendum rather than as part of the original Recommendation*.

Note – A PICS for a specific protocol implementation will need to be accompanied by administrative and documentary information relating to the supplier, the system and its intended environment.

7.1.2 The PICS proforma shall be in the form of a questionnaire or checklist to be completed by the supplier or implementor of an implementation of the relevant OSI* protocol.

7.1.3 The PICS proforma shall cover all functions, elements of procedure, parameters, options, PDUs, timers, multi-layer dependencies and other capabilities identified in the protocol Recommendation*, including optional and conditional capabilities. It is desirable, where practicable, to include all mandatory features. There shall be a well-defined mapping between static conformance requirements and the PICS proforma and there shall be consistency between them.

7.1.4 The PICS proforma shall be preceded by a section that states:

> "The supplier of a protocol implementation which is claimed to conform to this Recommendation shall complete the following Protocol Implementation Conformance Statements (PICS) proforma and shall provide the information necessary to identify uniquely both the supplier and the implementation."

7.1.5 The name, version and date of the relevant protocol Recommendation* shall be stated on each page of the PICS proforma.

7.1.6 The PICS proforma for a specific protocol shall contain:

a) explanations of special symbols, abbreviations, special terms, together with appropriate references;

b) explicit instructions for completing and interpreting the PICS;

c) provision for mentioning any mandatory feature that has not been implemented, with the appropriate rationale;

d) one or more tables (or other kinds of form as necessary) to be completed to state the capabilities of the implementation in detail, including:

   1) name of the feature, PDU type, timer, parameter, and other capabilities;

   2) a column stating whether each is mandatory, optional, negotiable, or conditional;

   3) wherever feasible, a column giving references to the relevant sections in the Recommendation;

   4) a column giving the permitted range or values, if appropriate;

   5) a column to be filled in to give the supported values or range of values, if appropriate;

   6) a column to be filled in to state if each capability has been implemented;

e) the proforma shall give a clear indication of the preferred data types (e.g. number bases, string types, octets, bits, seconds, minutes, etc.) for responses.

7.1.7 The PICS proforma shall use the following abbreviations as appropriate, unless they conflict with the abbreviations used in the specific protocol Recommendation*:

m	mandatory
o	optional
c	conditional
n	negotiable (using the protocol)
x	exclusion of capability
–	not applicable
s	ability to send
r	ability to receive

7.2 *Guidance on PICS proformas*

Appendix III provides some general purpose examples to give guidance on the construction of PICS proformas.

8 Test suite production process

In order to present the requirements and general guidance for abstract test suite specification, it is useful to assume a normal form of the process of test suite production. This section describes the process in just such a normal form. Abstract test suite specifiers are not required to follow this normal form exactly, however, they are recommended to use a similar process involving the same steps, possibly in a different order.

For the purposes of this Recommendation, the abstract test suite production process is assumed to be as follows:

a) study the relevant Recommendation(s)* to determine what the conformance requirements (including options) are which need to be tested, and what needs to be stated in the PICS (see § 9);

b) decide on the test groups which will be needed to achieve the appropriate coverage of the conformance requirements and refine the test groups into sets of test purposes (see § 10);

c) specify generic test cases for each test purpose, using some appropriate test notation (see § 11);

d) choose the test method(s) for which the complete abstract test cases need to be specified, and decide what restrictions need to be placed on the capabilities of the lower tester and [if appropriate to the chosen test method(s)] the upper tester and test coordination procedures (see § 12);

e) choose the test notation for specifying the complete abstract test cases, and specify the complete abstract test cases, including the test step structure to be used (see § 13);

f) specify the inter-relationships among the test cases and those between the test cases and the PICS and, as far as possible, the PIXIT, in order to determine the restrictions on the selection and parameterization of test cases for execution, and on the order in which they can be executed (see § 14);

g) consider the procedures for maintaining the abstract test suite (see § 15).

The remainder of this section provides requirements and guidance which relate to each step in the above process.

9 Determining the conformance requirements and PICS

9.1 *Introduction*

Before an abstract test suite can be specified, the test suite specifier shall first determine what the conformance requirements are for the relevant Recommendation(s)* and what is stated in the PICS proforma concerning the implementation of those Recommendation(s)*.

9.2 *Conformance requirements*

Section 1 of this part of the Recommendation specifies the requirements to be met by protocol specifiers as a prerequisite to the production of an abstract test suite for a particular protocol.

In practice, early OSI* Recommendations* might not contain a clear specification of all the relevant conformance requirements. In particular, the static conformance requirements might be badly specified or even omitted. In such cases, the test suite specifier shall contribute to the production of an amendment or addendum to the relevant Recommendation(s)* to clarify the conformance requirements. If, however, an abstract test suite has to be produced in advance of the conformance requirements being clarified in the relevant Recommendation(s)*, then the test suite specifier shall adopt the short-term solution given in Annex C and state clearly in the test suite Recommendation* what the implications of this are (i.e. what is assumed to be mandatory, what conditional and what the conditions are, and what is assumed to be optional).

9.3 *PICS proforma*

If no PICS proforma is included in a relevant protocol Recommendation*, the test suite specifier shall provide a PICS proforma to be processed as an addendum to that Recommendation*, or in exceptional circumstances (as discussed in § 7.1.1) as an annex to the abstract test suite Recommendation*.

Note – Progressing an addendum to an existing protocol Recommendation* may well be faster than progressing the abstract test suite Recommendation* because the PICS proforma is likely to be less controversial than the test suite and therefore is likely to require fewer revisions before a final text can be agreed.

10 Test suite structure

10.1 *Basic requirements*

An abstract test suite shall comprise a number of test cases. The test cases shall be grouped into test groups, if necessary nested. The structure shall be hierarchical in the sense that an item at a lower level shall be completely contained within a higher level item. The structure need not, however, be strictly hierarchical in the sense that any one test case may occur in more than one test suite or test group. Similar test groups may occur in more than one higher level test group or test suite.

The abstract test suite specifier shall ensure that a subset of the test purposes of each abstract test suite is concerned with capability testing, and another subset is concerned with behaviour testing. This need not lead to distinct test cases for behaviour and capability testing because it may be possible to use the same test steps for both a behaviour test purpose and for a capability test purpose. The test suite specifier shall provide an explanation of how the test purposes are derived from or relate to the protocol Recommendation*. The test suite specifier shall also provide a summary of the coverage achieved by the test suite.

10.2 *The test group structure*

In order to ensure that the resulting abstract test suite provides adequate coverage of the relevant conformance requirements, the test suite specifier is advised to design the test suite structure in terms of nested test groups in a top down manner (see Figure 1/X.290, Part 2).

There are many ways of structuring the same test suite into test groups; no one way is necessarily right and the best approach for one test suite may not be appropriate for another test suite. Nevertheless, the test suite specifier shall ensure that the test suite includes test cases for whichever of the following categories is relevant:

a) capability tests (for static conformance requirements);

b) behaviour tests of valid behaviour;

c) behaviour tests of syntactically invalid behaviour;

d) behaviour tests of inopportune behaviour;

e) tests focusing on PDUs sent to the IUT;

f) tests focusing on PDUs received from the IUT;

g) tests focusing on interactions between what is sent and received;

h) tests related to each protocol mandatory feature;

i) tests related to each optional feature which is implemented;

j) tests related to each protocol phase;

k) variations in the test event occurring in a particular state;

l) timing and timer variations;

m) PDU encoding variations;

n) variations in values of individual parameters;

o) variations in combinations of parameter values.

This list is not exhaustive; additional categories might be needed to ensure adequate coverage of the relevant conformance requirements for a specific test suite. Furthermore, these categories overlap one another and it is the task of the test suite specifier to put them into an appropriate hierarchical structure. The following structure is an example of a single-layer test suite, provided for guidance:

A Capability tests

A.1 Mandatory features

A.2 Optional features said by the PICS to be supported

B Behaviour tests: response to valid behaviour by peer implementation

B.1 Connection establishment phase (if relevant)

B.1.1 Focus on what is sent to the IUT

B.1.1.1 Test event variation in each state
B.1.1.2 Timing/timer variation
B.1.1.3 Encoding variation
B.1.1.4 Individual parameter value variation
B.1.1.5 Combination of parameter values

B.1.2 Focus on what is received from the IUT

– substructured as B.1.1

B.1.3 Focus on interactions

– substructured as B.1.1

B.2 Data transfer phase

– substructured as B.1

B.3 Connection release phase (if relevant)

– substructured as B.1

C Behaviour tests: response to syntactically invalid behaviour by peer implementation

C.1 Connection establishment phase (if relevant)

C.1.1 Focus on what is sent to the IUT

C.1.1.1 Test event variation in each state
C.1.1.2 Encoding variation of the invalid event
C.1.1.3 Individual invalid parameter value variation
C.1.1.4 Invalid parameter value combination variation

C.1.2 Focus on what the IUT is requested to send

C.1.2.1 Individual invalid parameter values
C.1.2.2 Invalid combinations of parameter values

C.2 Data transfer phase

– substructured as C.1

C.3 Connection release phase (if relevant)

– substructured as C.1

D Behaviour tests: response to inopportune events by peer implementation

D.1 Connection establishment phase (if relevant)

D.1.1 Focus on what is sent to the IUT

D.1.1.1 Test event variation in each state
D.1.1.2 Timing/timer variation
D.1.1.3 Special encoding variations
D.1.1.4 Major individual parameter value variations
D.1.1.5 Variation in major combination of parameter values

D.1.2 Focus on what is requested to be sent by the IUT

– substructured as D.1.1

D.2 Data transfer phase

– substructured as D.1

D.3 Connection release phase (if relevant)

– substructured as D.1

If the test suite is to cover more than one layer, then a single-layer test suite structure such as this could be replicated for each layer concerned. In addition, a correspondingly detailed structure could be produced for testing the capabilities and behaviour of multiple layers taken as a whole, including the interaction between the activities in adjacent layers.

10.3 *Test purposes*

The test suite specifier shall ensure that, for each test case in an abstract test suite, there is a clear statement of the test purpose. It is suggested that these test purposes should be produced as the next refinement of the test suite, after its structure in terms of test groups has been defined. The test purposes could be produced directly from studying those sections in the relevant Recommendation(s)* which are appropriate to the test group concerned. For some test groups, the test purposes might be derivable directly from the protocol state table; for others, they might be derivable from the PDU encoding definitions or the descriptions of particular parameters, or from text which specifies the relevant conformance requirements. Alternatively, the test suite specifier could employ a formal description of the protocol(s) concerned and derive test purposes from that by means of some automated method.

Whatever method is used to derive the test purposes, the test suite specifier shall ensure that they provide an adequate coverage of the conformance requirements of the Recommendation(s)* concerned. There shall be at least one test purpose related to each distinct conformance requirement.

In addition, it is possible to give guidance on the meaning of "adequate coverage" with reference to the above example. In order to express this, a shorthand notation will be used: the letter "x" will represent all appropriate values for the first digit in the test group identifier, and similarly "y" for the second digit, so that B.x.y.1 stands for B.1.1.1, B.1.2.1, B.1.3.1, B.2.1.1, B.2.2.1, B.2.3.1, B.3.1.1, B.3.2.1 and B.3.3.1. With this notation, a minimum "adequate coverage" for the above example is considered to be as follows:

a) for capability test groups (A.1, A.2):

   1) at least one test purpose per relevant feature, class or subset;

   2) at least one test purpose per relevant PDU type and each major variation of each such type, using "normal" or default values for each parameter;

b) for test groups concerned with test event variation in each state (B.x.y.1, C.x.1.1, D.x.y.1):

   - at least one test purpose per relevant state/event combination;

c) for test groups concerned with timers and timing (B.x.y.2, D.x.y.2):

   1) at least one test purpose concerned with the expiry of each defined timer;

   2) at least one test purpose concerned with very rapid response for each relevant type of PDU;

   3) at least one test purpose concerned with very slow response for each relevant type of PDU;

d) for test groups concerned with encoding variations (B.x.y.3, C.x.1.2, D.x.y.3):

   - at least one test purpose for each relevant kind of encoding variation per relevant PDU type;

e) for test groups concerned with valid individual parameter values (B.x.y.4, D.x.y.4):

   1) for each relevant integer parameter, test purposes concerned with the boundary values and one randomly selected mid-range value;

   2) for each relevant bitwise parameter, test purposes for as many values as practical, but not less than all the "normal" or common values;

   3) for other relevant parameters, at least one test purpose concerned with a value different from what is considered "normal" or default in other test groups;

f) for test groups concerned with invalid individual parameter values (C.x.1.3, C.x.2.1):

   1) for each relevant integer parameter, test purposes concerned with invalid values adjacent to the allowed boundary values plus one other randomly selected invalid value;

   2) for each relevant bitwise parameter, test purposes for as many invalid values as practical;

   3) for all other relevant types of parameter, at least one test purpose per parameter;

g) for test groups concerned with combinations of parameter values (B.x.y.5, C.x.1.4, C.x.2.2, D.x.y.5):

   1) at least one test purpose per "critical" value pair;

   2) at least one test purpose per pair of interrelated parameters to test a random combination of relevant values.

11 Generic test case specification

11.1 *Introduction*

The test suite specifier is recommended to specify a generic test suite, particularly if there is an intention to produce more than one abstract test suite.

A generic test suite shall consist of one generic test case for each test purpose. Each generic test case will be a refinement of its test purpose, which can be used as a common root of corresponding abstract test cases for different test methods.

If a generic test suite is produced in advance of abstract test suites, then it will be a useful step in the design process. If the generic test suite is produced after the production of at least one abstract test suite, then it will provide a means of relating different test suites to one another and analyzing where there may be gaps in their coverage.

11.2 *Description of generic test cases*

A generic test case consists of a test description of an "initial state" [of the test body] and a specification of the test body in a standardized test notation. The "initial state" includes not only the protocol state, but also any necessary information concerning the state of the SUT and the testing environment.

The test body is that part of a test case in which verdicts related to the test purpose are assigned to foreseen outcomes. The test body:

a) shall be defined in the style of either the Remote or Distributed test method;

   Note – Generic test cases may use the full expressive power of these methods (see § 12.3.3 for details), including the use of abstract local primitives.

b) shall assign verdicts to all possible outcomes; all outcomes yielding "pass" verdicts shall be explicitly identified, whereas all outcomes yielding "fail" or "inconclusive" verdicts shall be either identified or categorized (which may include categorization of default verdicts);

c) shall be described using a standardized test notation; the Tree and Tabular Combined Notation (TTCN) (defined in Annex D) is recommended.

11.3 *Relation of generic to abstract test cases*

For a particular abstract test method there may be many abstract test cases that can be derived from a single generic test case. A major difference between an abstract test case and a generic test case is that the abstract test case includes specifications of a preamble and a postamble. The preamble starts from a chosen stable state and leads to the required initial state of the test body. The postamble starts from the end of the test body and returns to a chosen stable state.

The preamble and postamble may be realized in different ways depending on the degree of control and observation provided by the test method used, or on the variety of different possible stable states which the derived abstract test case can start from and end in. These abstract test cases are simply different ways of achieving the same test purpose.

In addition, the test body of an abstract test case may be different from the corresponding generic test case if the test method used for the abstract test case is different from that used for the generic test case.

If a generic test suite is produced, it shall be used as the means of relating corresponding abstract test suites for different abstract test methods.

12 Abstract test methods

12.1 *Introduction*

Each abstract test suite shall be specified in terms of the control and observation of events in accordance with one of the abstract test methods defined in this section. The chosen test method determines the points at which control and observation shall be specified and the categories of events which shall be used [e.g. (N – 1)-ASPs, (N)-PDUs].

12.2 *General specification of the abstract test methods*

12.2.1 *End-system IUTs*

For IUTs within end-system SUTs there are four categories of abstract test methods, one local, and three external ones which assume the lower tester is located remotely from the SUT and connected to it by a link or network.

For generality, the test methods are described with reference to an IUT in which the highest layer is numbered "N_t" (for "top") and in which the lowest layer protocol to be tested is numbered "N_b" (for "bottom"). The IUT may implement protocols in layers lower than "N_b", but these are not of interest in the test method descriptions. The description applies to single-layer IUTs by taking N_t to be equal to N_b.

12.2.2 *Abstract local primitives*

Abstract test suite specifiers may, if necessary, define a set of abstract local primitives (ALPs) to be used in specifying the test suite. An ALP is an abbreviation for a description of control and/or observation to be performed by the upper tester, which cannot be described in terms of ASPs but which initiate events or state changes defined within the relevant protocol Recommendation(s)*. ALPs may be used with any abstract test method for end-system IUTs, but, for simplicity, will not be shown in any of the figures which illustrate those methods.

Any test case which uses an ALP shall be optional in the abstract test suite.

12.2.3 *The local test methods*

Abbreviation: L

In this method:

a) the abstract test suite is specified in terms of control and observation of (N_b – 1)-ASPs and (N_b) to (N_t)-PDUs;

b) the abstract test suite is also specified in terms of control and observation of (N_t)-ASPs;

c) the abstract test suite may also be specified in terms of control and observation by the upper tester of abstract local primitives (ALPs);

d) the method requires access to the lower and upper boundaries of the IUT and a mapping between the specified ASPs and their realization within the SUT;

e) the requirements for the test coordination procedures are specified in the abstract test suite but no assumption is made regarding their realization;

f) the upper and lower testers are required to achieve control and observation of the specified ASPs and the required test coordination procedures. They are assumed to be integrated into the SUT.

This method is illustrated in Figure 1/X.290, Part 2.

12.2.4 *External test methods*

12.2.4.1 *The distributed test method*

Abbreviation: D

In this method:

a) the abstract test suite is specified in terms of control and observation of (N_b – 1)-ASP"s and (N_b) to (N_t)-PDUs;

b) the abstract test suite is also specified in terms of control and observation of (N_t)-ASPs; the method requires access to the upper boundary of the IUT and a mapping between the (N_t)-ASPs and their realization within the SUT;

c) the abstract test suite may also be specified in terms of control and observation by the upper tester of ALPs;

d) the requirements for the test coordination procedures are specified in the abstract test suite but no assumption is made regarding their realization;

e) the upper tester is required to achieve control and observation of the specified (N_t)-ASPs and to achieve the effects of the required test coordination procedures; no other assumptions are made;

f) the lower tester is required to achieve control and observation of specified (N_b)-ASP"s and specified PDUs and to achieve the required test coordination procedures.

This method is illustrated in Figure 2/X.290, Part 2.

12.2.4.2 *The coordinated test method*

Abbreviation: C

In this method:

a) the abstract test suite is specified in terms of control and observation of $(N_b - 1)$-ASP"s, (N_b) to (N_t)-PDUs and test management PDUs (TM-PDUs);

b) (N_t)-ASPs need not be used in the specification of the abstract test suite; no assumption is made about the existence of an upper boundary of the IUT;

c) the requirements for the test coordination procedures are specified in the abstract test suite by means of a standardized test management protocol;

d) TM-PDUs may be defined which correspond to ALPs;

e) the upper tester is required to implement the test management protocol and achieve the appropriate effects on the IUT;

f) the lower tester is required to achieve control and observation of specified (N_b)-ASP"s and specified PDUs (including TM-PDUs).

This method is illustrated in Figure 3/X.290, Part 2.

12.2.4.3 *The remote test method*

Abbreviation: R

In this method, provision is made for the case where it is not possible to observe and control the upper boundary of the IUT. Also in this method:

a) the abstract test suite is specified in terms of control and observation of $(N_b - 1)$-ASP"s and (N_b) to (N_t)-PDUs ;

b) (N_t)-ASPs are not used in the specification of the abstract test suite; no assumption is made about the existence of an upper boundary of the IUT;

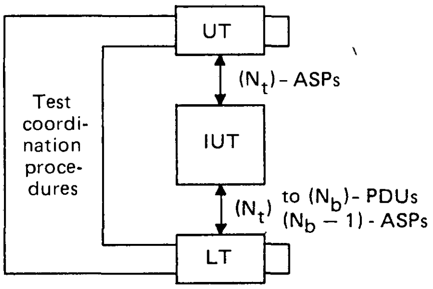
c) the abstract test suite may also be described in terms of control and observation of ALPs within the SUT;

d) some requirements for test coordination procedures may be implied or informally expressed in the abstract test suite but no assumption is made regarding their feasibility or realization;

e) abstractly the SUT needs to carry out some upper tester functions to achieve whatever effects of test coordination procedures and whatever control and/or observation of the IUT are implied, informally expressed or described by ALPs in the abstract test suite for a given protocol; these functions are not specified nor are any assumptions made regarding their feasibility or realization;

f) the lower tester is required to achieve control and observation of specified (N_b)-ASP"s and specified PDUs and should attempt to achieve the implied or informally expressed test coordination procedures in accordance with the relevant information in the PIXIT.

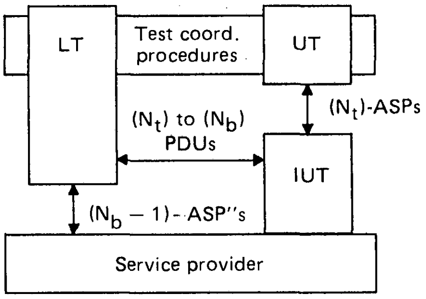
This method is illustrated in Figure 4/X.290, Part 2.



T0703730-88

FIGURE 1/X.290, Part 2

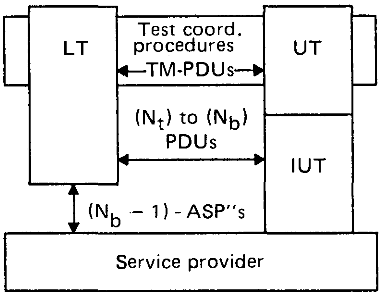
The local test method



T0703740-88

FIGURE 2/X.290, Part 2

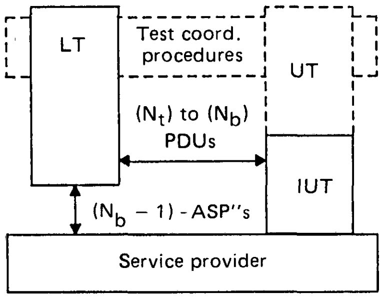
The distributed test method



T0703750-88

FIGURE 3/X.290, Part 2

The coordinated test method



T0703760-88

FIGURE 4/X.290, Part 2

The remote test method

12.2.5 *Single-layer, multi-layer and embedded variants*

Each category of test methods has a variant which can be applied to single-layer IUTs (abbreviation: S), and another which can be applied to multi-layer IUTs (abbreviation: M), when the set of adjacent layers is to be tested in combination (as a whole).

For a multi-layer IUT in which the protocols are to be tested layer by layer, an embedded variant of the test methods has been defined (abbreviation: E).

If control and observation are applied to a means of access to the upper boundary of the entities under test within SUT, then the test methods are normal (and no E is added to the abbreviated name). If, however, control and observation are applied through one or more OSI* layer entities above the entities under test, the test methods are called embedded (and an E is appended to the abbreviated name).

Names of particular variants of the test methods shall be formed as follows:

$$\left| \begin{array}{c} L \\ R \\ C \\ D \end{array} \right| \quad \left| \begin{array}{c} S \\ M \end{array} \right| \quad [\ E\ ]$$

For example, DSE is the abbreviation for the "distributed single embedded" test method.

12.2.6 *Open relay-systems*

For open relay-systems, loop-back and transverse abstract test methods are defined. They are given the abbreviated names: YL and YT, respectively.

12.3 *Single-layer test methods for single-layer IUTs in end-systems*

For single-layer test methods, the abstract model of the IUT is called the (N)-entity under test.

12.3.1 *The local single-layer test method*

The Local Single-layer (LS) abstract test method defines the points of control and observation as being at the service boundaries above and below the (N)-entity under test. The test events are specified in terms of (N)-ASPs above the IUT, and (N – 1)-ASPs and (N)-PDUs below the IUT, as shown in Figure 5/X.290, Part 2. In addition, ALPs may be used as test events. Abstractly, a lower tester is considered to observe and control the (N – 1)-ASPs and (N)-PDUs while an upper tester observes and controls the (N)-ASPs and ALPs. The requirements to be met by test coordination procedures used to coordinate the realizations of the upper and lower testers are defined in the abstract test suites, although the test coordination procedures themselves are not.

12.3.2 *The distributed single-layer test method*

The Distributed Single-layer (DS) abstract test method defines the points of control and observation as being at the service boundaries above the (N)-entity under test and above the (N – 1)-Service Provider at the SAP remote from the (N)-entity under test. The test events are specified in terms of (N)-ASPs above the IUT and (N – 1)-ASP"s and (N)-PDUs remotely, as shown in Figure 6/X.290, Part 2. In addition, ALPs may be used as test events. Abstractly, lower and upper testers are again considered to observe and control the behaviour at the respective points. The requirements to be met by the test coordination procedures are again defined in the abstract test suites, although the procedures themselves are not.

For lower layers (1-3) where it may be unrealistic to specify observation and control of (N – 1)-ASP"s, the lower tester observation and control shall be specified in terms of (N)-PDUs and, when necessary, changes in the state of the underlying connection.

Note – For example, the state of the underlying connection could be changed by setting up a new connection, or resetting or closing an existing connection.

The observation and control to be performed by the lower tester can optionally be specified in terms of (N)-ASP"s where this will reduce the size of the test case specification without loss of required precision.

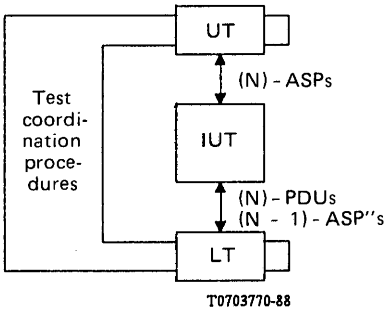


FIGURE 5/X.290, Part 2

The LS test method

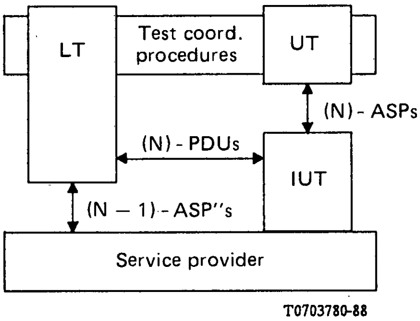


FIGURE 6/X.290, Part 2

The DS test method

12.3.3 *The coordinated single-layer test method*

The Coordinated Single-layer (CS) abstract test method is an enhanced version of the DS method, using a standardized upper tester and the definition of a test management protocol to realize the test coordination procedures between the upper and lower testers. The same standardized upper tester and test management protocol are not necessarily applicable to all test suites which use the coordinated method.

Standardized upper testers and test management protocols are applicable to a particular standardized abstract test suite for the coordinated test method and may not be applicable to other abstract test suites for the coordinated method.

There is only a single point of control and observation, above the (N – 1)-Service Provider at the SAP remote from the (N)-entity under test. Test events are specified in terms of (N – 1)-ASP"s, (N)-PDUs and TM-PDUs, as shown in Figure 7/X.290, Part 2.

For lower layers (1-3) where it may be unrealistic to specify observation and control of (N – 1)-ASP"s, the observation and control shall be specified in terms of TM-PDUs, (N)-PDUs and, when necessary, changes in the state of the underlying connection.

Concerning the test management protocol:

a) the test management protocol shall be implemented within the SUT directly above the abstract service boundary at the top of the IUT;

b) the IUT shall not be required to interpret TM-PDUs, only pass them to and from the upper tester;

c) a test management protocol is only defined for testing a particular protocol and so does not need to be independent of the underlying protocol;

d) verdicts on test cases shall not be based on the ability of the SUT to exhibit any ASP or parameter of an ASP at the upper service boundary of the IUT, since this would contradict the definition of the coordinated method in that the upper service boundary of the IUT is not a point of control and observation in this method. However, it is recommended that the test management protocol be defined separately from the abstract test suite(s) in order to facilitate the task of the implementor of an upper tester. This definition (as with the definition of any OSI* protocol defined by ISO or CCITT) can refer to the ASPs of its underlying service (i.e. the ASPs at the upper service boundary of the IUT);

e) TM-PDUs which correspond to ALPs are optional and there is no requirement for the upper tester to support them.

12.3.4 *The remote single-layer test method*

The Remote Single-layer (RS) abstract test method defines the point of control and observation as being above the (N – 1)-Service Provider at the SAP remote from the (N)-entity under test. The test events are specified in terms of the (N – 1)-ASP"s and (N)-PDUs remotely, as shown in Figure 8/X.290, Part 2. In addition, ALPs may be used as test events. Some requirements for test coordination procedures may be implied or informally expressed in the abstract test suites, but no assumptions shall be made regarding their feasibility or realization.

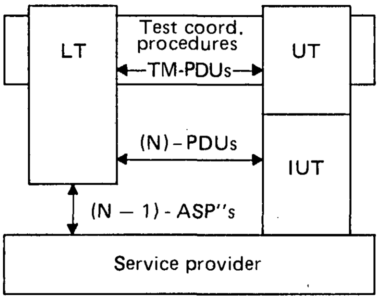
For the lower layers (1-3) where it is unrealistic to specify observation and control of (N – 1)-ASP"s, the observation and control shall be specified in terms of (N)-PDUs and when necessary the state of the underlying connection.

In addition, ın order to overcome the lack of specification of behaviour above the (N)-entity under test, where necessary the required behaviour of the system under test shall be specified in terms of the (N – 1)-ASP"s or (N)-PDUs which need to be observed by the lower tester. This form of implicit specification shall be taken to mean "do whatever is necessary within the system under test in order to provoke this required behaviour".

Note – Such implicit specification is necessary with this test method for any test case which requires an IUT initiated event which cannot be initiated by means of an ALP. Since ALPs may only be defined if the same effect cannot be achieved by ASPs, then any PDU which can be initiated by an ASP needs implicit specification to allow it to be initiated in this test method.

However, it is possible that some of the test cases in the abstract test suite cannot be executed (e.g. transmission and maintenance of busy conditions, transmission of consecutive unacknowledged Data PDUs, etc.). In such instances, it is left to the test laboratory and client to negotiate the method by which these tests can be accomplished.

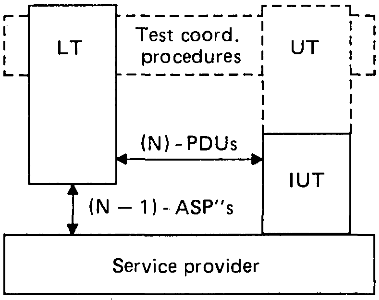
Even with such implicit specification of control of the IUT, in this method it is possible to specify control but not observation above the IUT, except through the use of ALPs. This is a major difference between this and the DS and CS test methods.



T0703790-88

FIGURE 7/X.290, Part 2

The CS test method



T0703800-88

FIGURE 8/X.290, Part 2

The RS test method

12.4 *Multi-layer test methods for multi-layer IUTs (LM, DM, CM, RM)*

Multi-layer testing, when the combined allowed behaviour of the multi-layer implementation is known, involves testing all the layers of a multi-layer IUT as a whole, without controlling or observing any of the inter-layer boundaries within the IUT.

In the Local Multi-layer (LM) test method the points of observation and control are the service boundaries directly above and below the IUT. The test events shall be specified in terms of the (N_t)-ASPs and ALPs above the IUT and the (N – 1)-ASPs and (N) to (N_t)-PDUs below the IUT.

In the Distributed Multi-layer (DM) test method the points of observation and control are at the service boundary above the IUT and above the (N – 1)-Service Provider at the SAP remote from the IUT. The test events shall be specified in terms of the (N_t)-ASPs and ALPs above the IUT and the (N – 1)-ASP"s and (N) to (N_t)-PDUs remotely.

In the Coordinated Multi-layer (CM) test method the point of observation and control is above the (N – 1)-Service Provider at the SAP remote from the IUT. The test events shall be specified in terms of the (N – 1)-ASP"s, the (N) to (N_t)-PDUs and the TM-PDUs. The test management protocol shall be designed to operate over the (N_t)-Service, where (N_t) is the highest layer in the IUT.

In the Remote Multi-layer (RM) test method the point of observation and control is above the (N – 1)-Service Provider at the SAP remote from the IUT. The test events shall be specified in terms of the (N – 1)-ASP"s and the (N) to (N_t)-PDUs, ALPs and the implicit specification of the control of the SUT behaviour when necessary. Some requirements for test coordination procedures may be implied or informally expressed, but no assumptions shall be made regarding their feasibility or realization.

12.5 *Single-layer testing of multi-layer IUTs or SUTs (Embedded methods)*

In embedded single-layer test methods testing is specified for a single-layer within a multi-layer IUT, including the specification of the protocol activity in the layers above the one being tested but without specifying control or observation at service boundaries within the multi-layer IUT. Thus in a multi-layer IUT from layer (N) to (N_t), abstract test cases for testing layer (N_i) shall include the specification of the PDUs in layers (N_i+1) to (N_t) as well as those in layer (N_i).

The Local Single-layer Embedded (LSE) test method uses the same points of control and observation as the LM test method for the same set of layers. The test events shall also be specified in the same terms as for the LM test method. The difference is that the LSE test method concentrates on a single-layer at a time, whereas the LM test method tests the multi-layer IUT as a whole.

In the Distributed Single-layer Embedded (DSE) test method for layer (N_i) within a multi-layer IUT from layer (N) to (N_t) the points of observation and control are at the service boundary above the IUT and above the (N_i – 1)-Service Provider at the SAP remote from the IUT, as illustrated in Figure 9a)/X.290, Part 2. The test events shall be specified in terms of (N_t)-ASPs and ALPs above the IUT and (N_i – 1)-ASP"s and (N_i) to (N_t)-PDUs remotely.

Note – For the top layer in the multi-layer IUT, (N_t), this method is the same as the DS test method.

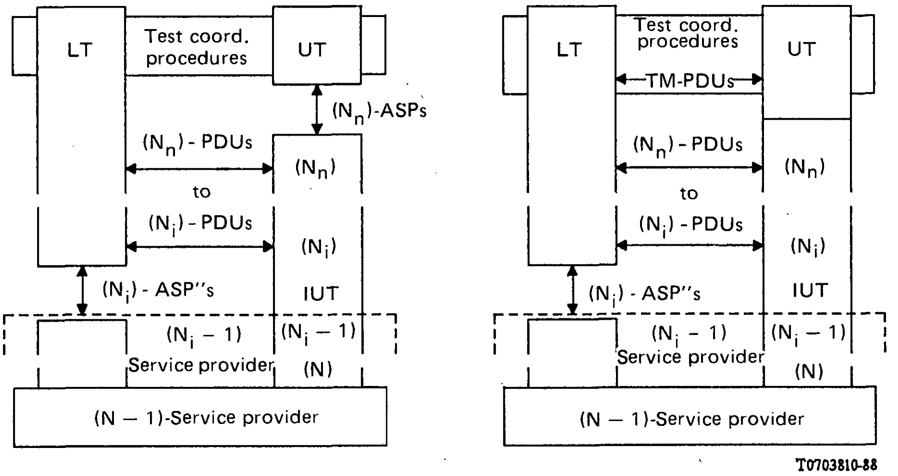
The Coordinated Single-layer Embedded (CSE) test method uses features of both the CM and DSE test methods. The test events shall be specified in terms of (N_i – 1)-ASP"s, (N_i) to (N_t)-PDUs and TM-PDUs and the test management protocol shall be designed to operate over the (N_t)-Service. This is illustrated in Figure 9b)/X.290, Part 2.

The Remote Single-layer Embedded (RSE) test method uses the same point of control and observation as the RS test method for the same layer, but differs from the RS test method in that (N_i+1) to (N_t)-PDUs shall be specified in test cases for layer (N_i).

Successive use of a single-layer embedded test method (from layer (N) to (N_t)) is called incremental testing of a multi-layer IUT.

The DSE/CSE/RSE methods are defined for a single layer under test in a multi-layer IUT. This does not mean that there cannot be accessible service boundaries within the multi-layer IUT, just that no such boundaries are used in the test methods. Thus, all layers between the layer under test and the highest layer for which PDUs are expressed as test events in the abstract test suite shall be regarded as being part of the multi-layer IUT.

Note – DME/CME/RME test methods could theoretically be defined to test multiple layers as a whole within a larger multi-layer IUT, but in order to avoid excess complexity, they are not detailed in this Recommendation.



a) Example of DSE test method: testing (N_i)-protocol in (N) to (N_n)-layer IUT

b) Example of CSE test method: testing (N_i)-protocol in (N) to (N_n)-layer IUT

FIGURE 9/X.290, Part 2

The embedded test methods

12.6 *Relay test methods*

There are two abstract test methods for relay system testing:

a) the loop-back test method (YL): used for testing a relay system from one subnetwork.

This test method is illustrated in Figure 10/X.290, Part 2.

For this test method there are two points of observation and control on one subnetwork at SAPs remote from the (N)-Relay. For connection-oriented protocols, it requires that the two test connections are looped together on the far side of the (N)-Relay, but it is not specified whether this looping is performed within the (N)-Relay or in the second subnetwork. For connectionless protocols, it requires that the PDUs are looped back within the second subnetwork and addressed to return to the second point of observation and control.

b) the transverse test method (YT): used for testing a relay system from two subnetworks.

This test method is illustrated in Figure 11/X.290, Part 2.

In this test method there are two points of observation and control, one on each subnetwork, at SAPs remote from the (N)-Relay.

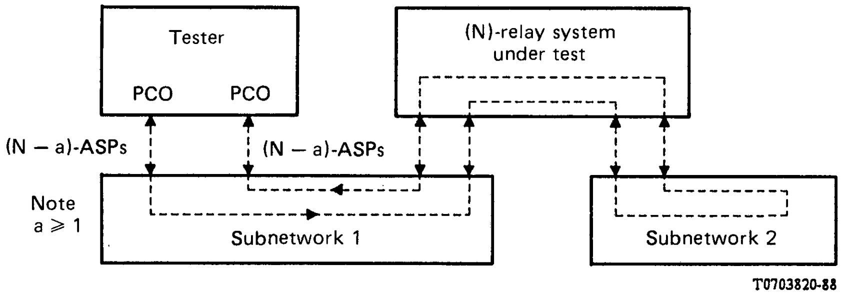


FIGURE 10/X.290, Part 2

Loop-back test method (YL)

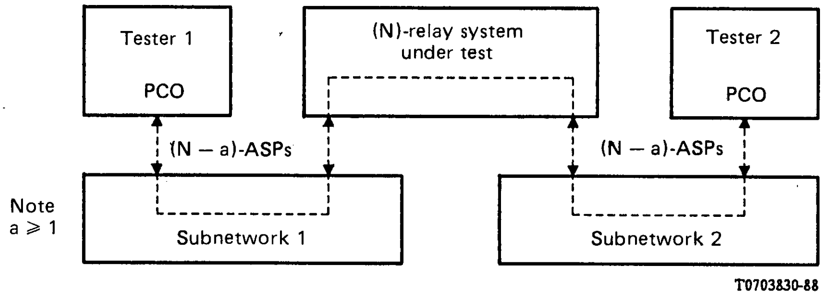


FIGURE 11/X.290, Part 2

Transverse test method (YT)

12.7 *Choice of test method*

12.7.1 *Introduction*

Before an abstract test suite can be defined, it is necessary to study all the environments in which the protocol is likely to be tested and establish accordingly the abstract test method(s) to be used for the production of one or more abstract test suites.

Abstract test suite specifiers shall place a requirement in the abstract test suite Recommendation* defining which abstract test method(s) shall be supported as a minimum by any organization claiming to provide a comprehensive testing service for the protocol(s) in question.

12.7.2 *IUT environments*

There is a relation between the test methods and the configurations of the real open systems to be tested.

Section 7.2 Part 1 of this Recommendation gives a full account of the classification of systems and IUTs.

When choosing a test method, the test suite specifiers should identify, if they have not already done so, whether the test suites they plan to produce are for IUTs which

a) are single or multi-layer;

b) belong to end or relay systems;

c) belong to complete or partial systems;

d) belong to fully open or mixed systems;

e) have service boundaries accessible or not;

f) are special purpose (i.e. to be used by a single application) or general purpose (i.e. to be used by several applications).

12.7.3 *Applicability of the abstract test methods*

The possibility of developing an abstract test suite for a given abstract test method will depend on the protocol(s) being considered, together with the results of the study described in § 12.7.2. This applies to the possibility of developing test suites for a given combination of methods (e.g. incremental) or a given variant of a method (e.g. embedded).

Some considerations concerning the applicability of the methods to different layers are discussed in Appendix I of Part 1 of this Recommendation.

One or more appropriate abstract test methods shall be selected for the protocol being considered.

Priorities should be assigned to the various test methods which have been identified, according to the configurations which are most likely to be found in real systems.

12.7.4 *Illustrative examples*

Appendix III provides an illustrative example of the choice of abstract test methods for given protocols.

12.8 *Test coordination procedures*

For effective and reliable execution of conformance tests, some set of rules is required for the coordination of the test process between the lower tester and the upper tester. The general objective of these rules is to enable the lower tester to control remotely the operation of the upper tester or vice versa, in ways necessary to run the test suite selected for the IUT.

These rules lead to the development of test coordination procedures to achieve the synchronization between the lower tester and the upper tester and the management of information exchanged during the testing process. The details and how these effects are achieved are closely related to the characteristics of the SUT, as well as the external test methods.

For each abstract test suite using the coordinated, distributed or local test methods, a standard set of test coordination procedures should be developed. This is because those procedures depend on the functionality of the upper tester and definitions of test cases.

In the case of the coordinated test methods (CS, CSE, CM) the test coordination procedures are realized by the standardization of a test management protocol. The test management protocol needs to be able to convey requests to the IUT to achieve the effect of a service primitive or ALP and to convey back to the lower tester the record of observations of effects equivalent to the occurrence of service primitives or ALPs. The upper tester should be an implementation of the test management protocol. It will be necessary to add test cases to the abstract test suite for the purpose of testing that the upper tester conforms to the requirements of the test management protocol specification. Such test cases do not contribute to the conformance assessment of the IUT.

When defining test cases for the local and distributed test methods, the test suite specifier shall record any constraints on the upper tester and/or test coordination procedures which may be necessary.

13 Specification of abstract test suites

13.1 *Test cases*

13.1.1 The abstract test suite specifier shall select an appropriate standardized notation in which to define the abstract test cases. The Tree and Tabular Combined Notation (TTCN), defined in Annex D, is defined for this purpose.

13.1.2 Once the test notation and test method have been chosen, then the generic test cases can be expanded into abstract test cases. There are two main kinds of change required to convert a generic test case into an abstract test case. The first is to express the test body in terms of control and observation required by the test method, and, if relevant, include a description of the synchronization needed between upper and lower testers. The second kind of change is to specify the preamble and postamble.

13.1.3 Specification of preambles and postambles may result in more than one test step for each. The preamble shall start in a stable state and progress to the desired state. Conversely, the postamble shall progress from the final state of the test body to a stable state. A small number of stable states shall be defined for the protocol concerned, always containing as a minimum the “idle” state. Each abstract test case shall be capable of being run on its own and shall therefore include test steps to start the preamble from the “idle” state and to end the postamble in the “idle” state.

However, other stable starting and final states for an abstract test case can be useful when efficient concatenation of abstract test cases is required.

Furthermore, in designing the test step structure within the abstract test cases, the test suite specifier can benefit from using the same test steps in several abstract test cases.

13.1.4 In converting from generic test cases to abstract test cases, the test suite specifier shall ensure that the initial state for the test body is preserved, the pass paths through the test body are preserved and the assignment of verdicts to outcomes remains consistent.

In order to maintain consistency, the following conditional requirements apply when assigning verdicts to outcomes:

a) if the behaviour of the preamble and the postamble are valid then the verdict assigned to a particular outcome shall be the same as that assigned to the corresponding outcome in the generic test case;

b) if the preamble results in the initial state of the test body not being reached, although there is no invalid behaviour, then the verdict shall be inconclusive;

c) if the preamble results in the initial state of the test body not being reached, because of invalid behaviour, then the verdict shall be “fail but test purpose inconclusive” (“fail type 3”);

d) if the postamble exhibits invalid behaviour and the generic test case (or test body) verdict is “pass” or “inconclusive”, then the verdict shall be “fail but test purpose passed” (“fail type 2”) or “fail but test purpose inconclusive” (“fail type 3”), respectively;

e) if the generic test case (or test body) verdict is “fail” then invalid behaviour in the postamble shall not change the verdict (“fail type 1”).

13.1.5 The test suite specifier shall also ensure that each abstract test case explicitly identifies all the outcomes assigned the verdict "pass" and identifies or categorizes all the remaining foreseen outcomes, assigning to each individual outcome or category a verdict "fail" or "inconclusive". All unforeseen outcomes in the test body shall be assigned either:

a) the verdict "fail"; or

b) the verdict "inconclusive".

The test suite specifier shall ensure that the application of a) or b) is consistent throughout the abstract test suite. If a) is chosen, then any unforeseen outcome in the preamble shall be assigned the verdict "fail but test purpose inconclusive" ("fail type 3"), and any unforeseen outcome in the postamble shall be treated as a protocol violation, leading to the appropriate type of fail verdict.

If b) is chosen, then any unforeseen outcome in the preamble shall be assigned the verdict "fail but test purpose inconclusive" ("fail type 3"), and any unforeseen outcome in the postamble shall be assigned the appropriate type of fail verdict.

13.2 *Test suites*

An abstract test suite specification comprises a set of test cases and test steps. Preceding the test cases themselves shall be the following information:

a) abstract test suite name, date of origin and version number;

b) names (and version numbers) of the protocol Recommendation(s)* for which test cases are provided;

c) names (and version numbers) of the service Recommendation(s)* whose primitives are specified as being observed;

d) name (and version number) of the Recommendation* defining the test notation, or a reference to an annex for the same if it is not standardized elsewhere;

e) name of target test method;

f) description of the coverage of the test suite; for example, the functional subsets of the protocol Recommendation(s)* that are tested;

g) description of the structure of the test suite, in terms of test groups and their relationship to the protocol Recommendation(s)*;

h) description of the test coordination procedures (if applicable in the test method);

i) statement of which test cases are optional and which mandatory for conformance to the abstract test suite Recommendation*;

j) information to assist the test realizer and test laboratory in their use of the abstract test suite Recommendation* (see § 14).

14 Use of an abstract test suite specification

The abstract test suite specifier shall provide information in the abstract test suite Recommendation* to assist the test realizer and test laboratory in their uses of the test suite. This information shall include, but is not limited to, the following:

a) a mapping of the abstract test cases to the PICS proforma entries to determine whether or not each abstract test case is to be selected for the particular IUT; the mapping should be specified in a suitable notation for Boolean expressions;

b) a mapping of the abstract test cases to the PIXIT proforma entries, to the extent that they are known by the abstract test suite specifier, to parameterize the test suite for the particular IUT and to determine which selected test cases cannot be run with the particular IUT.

The test suite specifier shall define a partial PIXIT proforma. This shall contain a list of all the ALPs used in the test suite (or test management protocol) and a list of all parameters for which the test suite requires values. If any of the required parameter values will be found in the PICS, the PIXIT proforma entry for each such parameter shall reference the corresponding entry in the PICS proforma;

Note – Other aspects of the PIXIT proforma are for further study;

c) a list of the abstract test cases in the order that shall be used in the Protocol Conformance Test `Report (PCTR), together with any information which shall be preserved in the PCTR on the status of each test case; this is a contribution to the development of a PCTR proforma;

d) any restrictions that there may be on the order in which the test cases may be executed;

e) reference to the description of the test coordination procedures (if applicable in the chosen test method);

f) any necessary timing information.

15 Test suite maintenance

Once an abstract test suite has been specified and is in use, it can be expected that errors and omissions in it will be detected by those who are using the test suite. The abstract test suite specifier shall in such circumstances progress the updating of the test suite via the relevant rapid amendment procedures.

In addition, from time-to-time, changes will be made to the protocol Recommendation(s)* to which the test suite relates. The abstract test suite specifier shall ensure that the test suite is updated as soon as possible after changes to the relevant protocol Recommendation* have been ratified.

ANNEX A

(to Recommendation X.290, Part 1)

Options

A.1 Options are those items in a Recommendation* for which the implementor may make a choice regarding the item to suit the implementation.

A.2 Such a choice is not truly free. There are requirements which specify the conditions under which the option applies and the limitations of the choice.

Conversely, there may be mandatory or conditional requirements, or prohibitions, in a Recommendation* which are dependent on the choice made or on a combination of the choices already made.

A.3 The following are examples of options and associated requirements; the list is not exhaustive:

a) "Boolean" options: the option is "do or do not do"; the requirement is "if do, then do as specified."

b) Mutually exclusive options: the requirement is to do just one of *n* actions, the option is which of them to do.

c) Selectable options: the option is to do any *m* out of *n* actions, with a requirement to do at least one action ($1 \leqslant m \leqslant n$ and $n \geqslant 2$).

A.4 Options may apply to anything within the scope of a Recommendation* (e.g. static or dynamic aspects, use or provision of a service, actions to be taken, presence/absence or form of parameters, etc.).

A.5 Another type of distinction is between service-user options and service-provider options.

A.6 In a wider context, the choice will be determined by conditions which lie outside the scope of the Recommendation* (e.g. other Recommendations* which apply to the implementation, the protocols used in the (N+1) and (N − 1) layers, the intended application, conditions of procurement, target price for the implementation, etc.). However, these have no bearing on conformance to the Recommendation* in which the option appears.

ANNEX B

(to Recommendation X.290, Part 2)

Guidance for protocol Recommendations* writers to facilitate conformance testing

B.1 Introduction

This Annex gives guidance, primarily for the specifiers of new protocol Recommendations*, to facilitate conformance testing by ensuring a very clear understanding of the conformance requirements.

B.2 *Guidance on scope and field of application*

B.2.1 Precision in the scope and field of application sections sets the tone for precision in the rest of the Recommendation*. The requirements stated in the Recommendation* should be consistent with the scope and field of application and *vice versa.*

B.2.2 The scope should distinguish clearly between the following three types of information included in the protocol Recommendation*:

a) the definition of the procedures for communication to be followed at the time of communication;

b) requirements to be met by suppliers of implementations of the procedures;

c) guidance on how to implement the procedures.

Guidance on how to implement the procedures does not constitute additional requirements nor does it have any bearing on conformance. If such guidance is included, the scope should make these points and indicate how guidance can be distinguished from the requirements of the Recommendation*. This distinction is much easier to make if guidance is separated from requirements. The recommended method of such separation in the ISO Directives is to place guidance in notes and annexes.

B.2.3 It should be clear to whom the Recommendation* applies.

B.2.4 It should be clear at what time the Recommendation* applies.

Protocol procedures apply between pairs of communicating parties at the time of communication. If there might be any ambiguity over which communicating parties are involved, this should be resolved in the scope.

It is best if protocol Recommendations* are written in such a way that the requirements are to be met by a single communicating party (the 'first' communicating party for this purpose) for the benefit of one or more other communicating parties (the 'second' communicating parties). Then when two (or more) communicating parties are all expected to communicate in conformance with the Recommendation*, the Recommendation* is first applied to one party, treating it as the 'first' one, and then to the other(s) in turn. This ensures that if the procedures are violated, it is clear which party is at fault.

B.2.5 If any guidance is given about factors which are not standardized definitively, the scope should make it clear that any such guidance can be ignored without affecting conformance.

B.2.6 The aspects which are excluded from the scope should be identified clearly.

Not all factors relevant to the procedures or to products which implement them need to be standardized; indeed it is often desirable to leave some implementor freedom. For instance, it may be desirable to omit in a protocol Recommendation* any requirements for explicit values of timeouts, but to give guidance instead.

The scope should make it clear which aspects are standardized definitively, which are covered by guidance but not by any requirements, and which are excluded from consideration by the Recommendation*. Any aspects which one might think would be covered, on the basis that they are closely related to aspects which are standardized, need explicit mention.

B.2.7 All options should, if possible, be identified clearly in the scope.

Options are one of the most troublesome, but unfortunately necessary parts of protocol Recommendations*. They fall somewhere between what is standardized and what is not. They will be covered in greater depth below. At this point, what is important is that options are not burried deep inside the Recommendation* but are declared clearly at the beginning. If the number and detailed nature of the options makes this impractical, one should seriously ask whether such complexity is really necessary. Can detailed options be grouped together in some way (e.g. classes) to simplify the Recommendation*?

B.2.8 The scope and field of application should be reviewed after considering the rest of the Recommendation*.

It is not often possible to satisfy some of the above suggestions until the rest of the Recommendation* has been considered. Therefore, it is generally necessary to return to the scope, to check that it really does agree with the contents of the Recommendation*. It is common to find that sections quite outside the scope have found their way in.

B.3 *Guidance on references*

B.3.1 OSI protocol Recommendations* should refer to the OSI reference model, the relevant service Recommendations* and to any relevant Recommendations* for protocol conventions, guidelines, or formal description techniques.

B.3.2 It should be made clear whether conformance to the protocol Recommendation* requires conformance to any part of any other Recommendantion*.

B.3.3 It should be made clear whether each reference is to a particular version of the reference Recommendation* or to each successive version.

Normally, the latest version is required, but this can cause problems as changes to the other Recommendation* might affect conformance to this Recommendation*.

B.4 *Guidance on requirements and options*

B.4.1 The status of every requirement should be unambiguous.

Since optional and conditional requirements are so common, there is a tendency to interpret everything which can be interpreted as optional as being optional.

B.4.2 It should be possible for an instance of communication to conform with all the mandatory dynamic conformance requirements.

B.4.3 The conditions under which conditional requirements apply should be spelt out clearly.

B.4.4 It should not be impossible for the implementor or supplier to know what these conditions are.

B.4.5 There should be no possibility of confusion between what is optional dynamically and what is optional statically.

There may be mandatory static conformance requirements for the support of features whose use at the time of communication is optional. Conversely, a message whose use is mandatory in a given context at the time of communication may be part of a protocol mechanism whose support is optional statically.

B.4.6 If the Recommendation* contains a 'shopping list' of options, and there are restrictions on the allowed combinations of such options, then the restrictions should be specified clearly. These should include identification of any mutual exclusions and any minimum and maximum limits to the allowed range of options.

B.4.7 If the Recommendation* does not give any rules for selection of options, it should be made clear in the scope that only the total range and individual options are standardized, but not the selection.

B.4.8 Legitimizing options should be avoided. These are options which allow alternative and incompatible versions of the same thing to claim conformance to the same Recommendation*. While they do not of themselves prevent an objective understanding of conformance, they may frustrate the aims of OSI.

B.4.9 There should be no options which give the implementor permission to ignore important requirements of the Recommendation*. Such options devalue the Recommendation* and the meaning of conformance to it.

B.4.10 If there are prohibitions in the Recommendation*, they should be sufficiently precise to be meaningful.

Many Recommendations* have sections which say in effect 'do all of this and nothing else'. Such prohibitions may be meaningless, because every protocol conveys some information which is not standardized, the so-called 'user data', and every standardized product has attributes which are not standardized, e.g. weight. It may be difficult to draw a clear objective line between things the Recommendation* cannot forbid and those which the writers of the Recommendation* want to forbid, unless the prohibitions are stated explicitly.

B.5 *Guidance on protocol data units*

B.5.1 The permitted set of PDU types and parameter encodings should be stated clearly.

B.5.2 The permitted range of values should be stated clearly for each parameter.

B.5.3 All values, which fall outside the stated permitted range, should be stated explicitly to be invalid.

If not, some people will argue that such values are undefined but allowed, whilst other will argue that they are invalid.

B.5.4 It should be clear whether or not undefined PDU types are allowed.

It is safer if all undefined PDU types are declared to be invalid.

B.5.5 Critical undefined values should be mentioned explicitly in the scope as being undefined.

B.5.6 There should be a defined procedure to be followed by the first communicating party in each case of it receiving an invalid or undefined PDU type or parameter.

B.5.7 It should be possible to detect whether the defined procedure has been followed in such cases. If it is not, then it should be because it does not matter.

Sometimes the procedure to be followed upon receipt of an invalid PDU is intentionally the same as when some valid PDUs are received in the same circumstances. For example, the procedure might be to do nothing until one specific type of PDU is received, everything else being ignored. In such cases, it probably does not matter that the error has apparently gone undetected. In some other cases, the intention may be that error cases should be given special treatment, but that the procedure has been poorly chosen with the result that it cannot be distinguished from the action in non-error cases.

B.5.8 If, in the encoding of PDUs, there are any fields declared to be reserved, then there should be a clear statement of what values, if any, are allowed or disallowed in these fields.

B.5.9 If inter-related parameters can be carried on separate PDUs, then the set of permitted relationships between the values of these parameters should be precisely and clearly defined.

B.5.10 If the parameter encoding allows for parameters to be specified in any order, and the PDU format places restrictions on the permitted orders, then these restrictions should be clearly stated. It should be recognized that if many different orders are permitted, then a large representative sample of different orders ought to be tested. The added complexity of testing conformance should, therefore, be adequately compensated by some advantage in allowing this freedom.

B.5.11 The order in which the bits, octets, etc. should be carried in the underlying protocol should be stated clearly.

For example, should a two octet integer travel most or least significant octet first? It is surprising how often such simple causes of ambiguity are overlooked.

B.5.12 The relationship between SDUs and PDUs should be defined clearly.

B.6 *Guidance on state*

B.6.1 Protocol procedures are often defined using a finite state approach, whether formalized or not. The specification of these states is often incomplete.

B.6.2 Each state should be defined clearly.

B.6.3 If there are events which can only occur in a subset of the possible states, then possible occurrence of an event should be distinguished from valid occurrence.

B.6.4 The required actions and state transitions should be defined for each possible state/event pair. In particular, they should be defined for possible but invalid state/event pairs.

B.7 *Guidance on formal description techniques*

B.7.1 The following requirements apply only to those Recommendations* which include a formal description. Precise, unambiguous Recommendations* can be written without the aid of a formal description technique (FDT), but in complex Recommendations* such as protocols formal descriptions are highly recommended. It should, however, be realized that they can create problems themselves in relation to conformance.

B.7.2 It should be clear whether the formal description forms an essential part of the Recommendation* or is provided only for guidance.

It is very important to have a clear understanding of the status of the formal description. Ideally there should be no discrepancies between the text and the formal description, but because this is very difficult to achieve in practice it is important that the reader knows which takes precedence. If the formal description is provided only for guidance, it cannot define conformance requirements.

B.7.3 The FDT should be well-defined, stable and properly referenced.

B.7.4 If the formal description defines requirements, but not all the requirements of the Recommendation*, then it should be stated clearly that the text includes requirements which are not covered by the formal description and these additional requirements should be identified clearly.

B.7.5 If the formal description defines requirements, and it also defines an allowed way of implementing some aspects of the protocol, but there is intended to be freedom for the implementor to implement those aspects in some other way, then this constitutes over-definition. This is all too common in formal descriptions, and creates difficulties in relation to conformance. If the formal description is an essential part of the Recommendation*, then text should be provided to qualify it, indicating where such over-definition exists and what the real requirements are.

The problem usually arises because the formal description describes the internal behaviour of an idealized implementation, rather than the observable external behaviour that is required. It is only the observable external behaviour which can be tested, and therefore it is only this which should constitute requirements for conformance purposes. It may well be that a different FDT should be used for defining the requirements from that used to provide guidance to implementors.

B.8 *Miscellaneous guidance*

B.8.1 Information which may appear obvious should nevertheless be stated.

If something is omitted because it is 'obvious', some readers will assume it is required because it is 'obvious', but others will assume that it is obmitted to provide freedom for implementors. For example, does the existence of a checksum imply that it has to be checked?

ANNEX C

(to Recommendation X.290, Part 2)

Incomplete static conformance requirements

C.1 As a matter of historical record, the development of protocol Recommendations* has been undertaken in parallel with the determination of the meaning of conformance, and in particular with the gaining of understanding of the distinction between static and dynamic conformance.

C.2 Therefore, some early protocol Recommendations* will not give a complete specification of the requirements for static conformance. Typical of the imcompleteness is the requirement to support a particular function without saying whether this applies to sending, receiving or both; or the absence of precise conditions of detection of protocol errors in received incoming messages.

C.3 Accordingly, there may be different interpretations of what a conforming implementation is.

C.4 In future Recommendations* or at the time of revision of existing Recommendations* it will be necessary to provide a thorough specification of static conformance requirements. This would include the specification of conditions applicable to the implementation or non-implementation of everything that is neither always mandatory nor always optional.

C.5 In the short term, it is essential that, at very least, current drafts are modified to clarify the present situation: it is not considered acceptable that Recommendations* should be issued in a form in which implementation requirements suffer from extensive ambiguity. Consideration also needs to be given to what should be done where the protocols have already reached the status of ISO International Standard or CCITT Recommendation.

C.6 No other short-term solution is seen other than to accept and state clearly that all capabilities not covered explicitly in the static conformance requirements are optional; and to minimize the potential problems that this may cause by specifying that:

a) only implementations which
   1) implement everything that is explicitly specified as mandatory; and
   2) do not omit anything unless it is explicitly stated to be optional, even though there may be a general clause of the sort 'if not specified, then optional';

   are to be designated as 'conforming' without qualification;

b) any implementation which
   1) implements everything that is explicitly specified as mandatory; and
   2) omits things which are not explicitly stated to be optional, perhaps because of a general clause of the sort 'if not specified then optional';

   shall be described as conforming to a subset.

C.7 Implementations which omit anything that is mandatory do not conform at all.

Note – A system conforming without qualification will not necessarily interwork with another system, nor will it necessarily work better than a system conforming to a subset. The 'perfect' system may refuse from other systems PDUs which it would consider as incorrect or incomplete. Thus, it may reject or abort connections.

Therefore, special consideration should be given to conformance with respect to the detection of protocol errors, especially when this detection can be explicitly or implicitly optional.

ANNEX D

(to Recommendation X.290, Part 2)

The tree and tabular combined notation

D.0 Introduction

In constructing an abstract test suite, a test notation is used to describe abstract test cases. The test notation can be an informal notation (without precisely defined semantics) or a formal description technique (FDT). This Annex describes an informal notation called the tree and tabular combined notation (TTCN).

TTCN serves the following purposes:

a) to provide a common reference for assessing other test notations and to assist in examining the problems arising in test case and test suite design;

b) to provide a basis for the translation of test cases into other test notations;

c) TTCN behaviour descriptions are appropriate for specifying test cases and test steps.

A test suite can be looked upon as a hierarchy ranging from the complete test suite down to test events (see Part 1, § 8.1). TTCN assumes that the basic types of test events are abstract service primitives (ASPs), abstract local primitives (ALPs) and timer events.

Abstract test cases can also be expressed in terms of PDUs (by using an abbreviation mechanism described in § D.5.11).

D.1 *Components of a TTCN test suite*

A test suite written in TTCN shall have the following four sections in the following order:

a) Suite overview (§ D.4)
Information needed for the general presentation and understanding of the test suite, such as test reference and a description of its overall purpose.

b) Declarations (§ D.5)
The alphabet of the events to be used in the test suite (e.g., SAPS, Timers, ASPs, PDUs, ALPs and their parameters) is described. This section shall contain the definition of any abbreviations to be used later in the test suite.

c) Dynamic part (§ D.6)
Tables containing trees of behaviour expressed mainly in terms of the occurrence of ASPs at points of control and observation. A set of main behaviour descriptions is followed by a set of default behaviour descriptions.

d) Constraints part (§ D.8)
This section specifies values for the ASPs, PDUs, and their parameters used in the Dynamic part.

D.2 *The syntax form of TTCN*

TTCN is provided in graphical form (TTCN-GR), suitable for human readability.

Note – A machine processable form of (TTCN-MP), suitable for transmission of TTCN descriptions between machine and possibly suitable for other automated processing is for further study.

D.3 *Conventions*

D.3.1 *Table proformas*

The TTCN-GR notation is defined using a number of different types of table. The following conventions will be used in the description of templates for these tables:

a) Bold text (**like this**), or text in this font shall appear verbatim in each actual table;

b) Text in italics (*this font*) shall not appear verbatim. This font is used to indicate that the actual text must be substituted for the italicised symbol.

Additionally, within table proformas comments may be placed in fields not reserved for commentary, by delimiting the comments with the symbols /* and */.

D.3.2 *Types of bracket*

The terms that will be used when referring to the various different types of bracket are defined in Figure D-2/X.290 Part 2.

Brackets: [. . .]
Braces: {. . .}
Parentheses: (. . .)

FIGURE D-2/X.290, Part 2

Brackets

D.3.3 *Naming conventions*

D.3.3.1 *Test group and test case references*

a) The reference (name) of a test group shall be of one of the forms shown in Figure D-3/X.290, Part 2 and illustrated in Example D-2.

(Test-suite)/(Test-group)/

(Test-suite)/(Test-group$_1$*)/. . ./(Test-group*$_n$*)/*

FIGURE D-3/X.290, Part 2

Test group references

Example D-2 – A transport group reference:

Transport/Class0/Conn-Estab/

b) The reference (name) of a test case shall be of one of the forms shown in Figure D-4/X.290, Part 2 and illustrated in Example D-3.

(TestSuite)/(TestCase)/

(TestGroupReference)/. . ./(TestCase)/

FIGURE D-4/X.290, Part 2

Test case references

Example D-3 – A transport test case reference:

```
Transport/Init
Transport/Class0/Conn-Estab/LT-Init
```

c) The test group references, together with the test case references define the structure of the test suite.

D.3.3.2 *Test step references*

a) Test steps are attached at a given point in the structure of the test suite, as defined by the group and case references (see § D.3.3.1).

b) Test steps, at their point of attachment may be grouped into "libraries", organized in hierarchical fashion.

c) A step reference is therefore of the form:

(PointOfAttachment) :(LibraryStructure)

where:

1) *(PointOfAttachment)* is either:

   A) *(TestSuite)* or

   B) *(TestGroupReference)* as defined in § D.3.3.1; or

   C) *(TestCaseReference)* as defined in § D.3.3.1;

2) and *(LibraryStructure)* is either:

   A) *(TestStep)* or

   B) *(Component)*$_1$)/. . ./ *(Component*$_n$*)/(TestStep)*

Example D-4 – Transport Test Step References:

```
Transport:Step-A
Transport/Class0/Conn-Estab/:Step-B
Transport/Class0/Conn-Estab/LT-Init:Step-C
Transport:Common-lib/Preambles/Step-C
```

Note 1 – The colon (:) in the test step reference separates the first part which refers to a point in the already defined test suite structure from the second part which defines the library structure.

Note 2 – The components allow the steps to be grouped into an arbitrary hierarchy within libraries, which have no influence on the test suite structure itself.

D.4 *Suite overview*

This section shall include at least the following information:

a) references to the relevant base standards;

b) a reference to the PICS and PIXIT and how they are used;

c) an indication of the test method or methods to which the test suite applies;

d) a complete index to the test suite, consisting of the test reference, test identifier, page number and test purpose for each test case and test step in the test suite. The test purposes shall be organized according to the structure of the test suite.

Other information which may aid understanding of the test suite, such as how it has been derived, should also be included as commentary.

This information shall be provided in the format shown in Figure D-5/X.290, Part 2.

D.5 *Declarations*

The purpose of the declarations section is to describe the set of test events and all other attributes to be used in the test suite. All objects used in the dynamic part shall have been declared in the declaration part. There are two sorts of test event:

a) abstract service primitives (ASPs) which occur at the points of control and observation (PCOs) used by the tester (§ D.5.8);

b) timer events (§ D.5.10).

Other attributes are also specified:

a) user defined types (§ D.5.1.3);

b) user defined operators (§ D.5.3);

c) test suite parameters (§ D.5.4);

d) global constants (§ D.5.5);

e) global variables (§ D.5.6);

f) PCOs (§ D.5.7);

g) ASP parameters (§ D.5.8);

h) data types (including PDUs and their parameters) (§ D.5.9);

i) abbreviations (§ D.5.11).

<table>
<tr><td colspan="4">Suite Overview</td></tr>
<tr><td colspan="4">Reference to Standards: . . .
Reference to PICS: . . .
Reference to PIXIT: . . .
How Used: . . .
Test Method(s): . . .
Comments: . . .</td></tr>
<tr><td>Test Case identifier</td><td>Test Case Reference</td><td>Page</td><td>Purpose</td></tr>
<tr><td>.</td><td>.</td><td>.</td><td>.</td></tr>
<tr><td>.</td><td>.</td><td>.</td><td>.</td></tr>
<tr><td>.</td><td>.</td><td>.</td><td>.</td></tr>
<tr><td><i>(Test Case Identifier)</i></td><td><i>(Test Case Reference)</i></td><td><i>(Page)</i></td><td><i>(Purpose)</i></td></tr>
<tr><td>.</td><td>.</td><td>.</td><td>.</td></tr>
<tr><td>.</td><td>.</td><td>.</td><td>.</td></tr>
<tr><td>.</td><td>.</td><td>.</td><td>.</td></tr>
</table>

FIGURE D-5/X.290, Part 2

Suite overview proforma

D.5.1 *General TTCN types*

TTCN supports both a number of predefined types and mechanisms that allow the definition of user declared types. These types may be used throughout the test suite and may be referenced when variables, constants, ASP parameters, PDU parameters, or test suite parameters are defined.

D.5.1.1 *Predefined types*

A number of commonly used types are predefined for use in TTCN. These types may be referenced even though they do not appear in a type declaration in a test suite. All other types used in a test suite must be named in the User Type Declarations (according to § D.5.1.2) and referenced by name.

a) **Integer Predefined Type**: a type with distinguished values wich are the positive and negative whole numbers, including zero (as a single distinguished value).

b) **Bitstring Predefined Type**: a type whose distinguished values are the ordered sequences of zero, one, or more bits.

c) **Octetstring Predefined Type**: a type whose distinguished values are the ordered sequences of zero, one, or more octet being an ordered sequence of eight bits.

d) **Character String Predefined Types**: types whose distinguished values are zero, one or more characters from some character set. The character string types listed in Figure D-6/X.290, Part 2 may be used. They are defined in section two of Recommendation X.208).

NumericString

PrintableString

TeletexString (T61String)

VideotexString

VisibleString (ISO646String)

IA5String

GraphicString

GeneralString

FIGURE D-6/X.290, Part 2

Predefined character string types

e) **The Connection Endpoint Identifier Predefined Type** (CEId): a type consisting of an unlimited number of distinguished values.

D.5.1.2 *User defined types*

Types specific to a test suit may be introduced by the TTCN user. To define a new type, the following information shall be provided:

a) a name for the type;

b) the base type (if any);

c) a definition of the type, provided in one of the following manners:

   1) by giving a precise reference to a clause, or clauses, of a standard which defines the type;

   2) by giving an ASN-1 [Recommendation X.208] type reference of the form:

      (modulereference).(typereference)

   3) by enumerating the set of named distinguished values comprising the type (see Figure D-7/X.290, Part 2);

   4) by specifying a subset of the distinguished values of another type.

      This may be done in a number of ways:

      A) by specifying a subrange of the *Integer* predefined type;

      B) by specifying a subrange of an enumerated type;

      C) by restricting the length of a *BitString*, *OctetString*, or a character string predefined type.

This information shall be provided in the format shown in Figure D-8/X.290, Part 2.

User Type Definition			
Name	Base Type	Definition	Comments

. . . *(Name)* . . .	. . . *(Base Type)* . . .	. . . *(Definition)* . . .	. . . *(Comments)* . . .

FIGURE D-7/X.290, Part 2

User type definition proforma

User Type Definitions			
Name	Base Type	Definition	Comments

Transport-Classes	None	(Class 0, Class 1, Class 2, Class 3, Class 4)	Classes that may be used for Transport layer connection

FIGURE D-8/X.290, Part 2

Example user type declaration

D.5.2 *Value denotation*

Values of the different types shall be denoted in the following fashion:

D.5.2.1 *Values of predefined types*

The values of predefined types shall be denoted as follows:

a) **Integer Values**: values of type *Integer* shall be denoted by one or more digits. The first digit shall not be zero unless the value is 0.

b) **BitString and OctetString Values**: values of type *BitString* and *OctetString* shall be denoted in one of the following ways:

1) by a list of bits;

In this case, the value shall be denoted by an arbitrary number (possibly zero) of zeros and ones, preceded by a single ' and followed by the pair of characters 'B.

Example D-5 – '01101100'B

2) by a list of semi-octets.

In this case, the value shall consist of an arbitrary number (possibly zero) of the characters

0 1 2 3 4 5 6 7 8 9 A B C D E F

preceded by a single ’ and followed by the pair of characters ’H. Each character is used to denote the value of a semi-octet using a hexadecimal representation.

Example D-6 – ’AB0196’H

c) **Character String Values**: values of character string types shall be denoted by an arbitrary number (possibly zero) of characters from the character set referenced by the character string type, preceded and followed by ”. If the character string type includes the character ”, this character shall be represented by a pair of ” in the denotation of any value.

d) **Connection Endpoint Identifier Values**: values of the *CEId* type have no denotation.

Note – None is required, since the only operation defined over the *CEId* type is equality (see § D.5.8).

D.5.2.2 *Values of user defined types*

Values of the user defined types shall be denoted as follows:

a) The denotation of values of the types introduced by reference to the section, or sections, of a Recommendation* shall specify how the distinguished values of that type are to be denoted in the test suite, in the comments associated with the type definition.

b) a value of a referenced ASN-1 type shall be denoted using ASN-1 either by

1) a reference of the form:

(modulereference).(valuereference) or

2) specifying an ASN-1 value of the given type.

Note – The ASN-1 modular method has extensions that allow the value to be partially specified (§ D.8.3).

c) a value of an enumerated type shall be denoted by its name.

d) a value of a type obtained by subsetting another type shall have the same denotation as the values of the type which was subset.

D.5.3 *Operators*

Operators specific to a test suite may be introduced by the TTCN user. To define a new operation, the following information shall be provided:

a) a name for the operation;

b) the signature of the operation, comprising:

1) a list of input types;

2) a name for each input component;

3) the type of the result;

c) a description of the operation.

This shall be provided in the format shown in Figure D-9/X.290, Part 2.

Operation Definition	
Operation Name:	*(Operation Name)*
Input Types:	List of *(Type Names)*
Components:	List of *(Name)*
Result Type:	*(Type Name)*

Description:	*(Description)*

FIGURE D-9/X.290, Part 2

Operation definition proforma

The definitions of two string operations are given in Figures D-10/X.290, Part 2 and D-11/X.290, Part 2.

Operation Definition	
Operation Name:	*substr*
Input Types:	*string x integer x integer*
Components:	*source, start, length*
Result:	*string*

Description:	*substr (source, start, length)* is the string of length *length* starting from index *start of source* For example: *substr ("abcde",3,2) = "cd"* *substr ("abcde",4,9999) = "de"*

FIGURE D-10/X.290, Part 2

Definition of operation substr

Operation Definition	
Operation Name:	*length*
Input Types:	*string*
Components:	*source*
Result:	*integer*

Description:	*length (source)* is the length of the string *source* For example: *length ("abcde") = "5"*

FIGURE D-11/X.290, Part 2

Definition of operation length

Note – An operation may be compared to a function in an ordinary programming language. However, the arguments to the operation are not altered as a result of any call of the operation – there are no side effects.

D.5.4 *Test suite parameters*

The purpose of this section is to declare constants derived from the PICS or PIXIT which globally parameterize the test suite. These constants are referred to as test suite parameters.

Note – In most real cases of testing Test Suite Parameters will be bound to a value when the PICS/PIXIT processing occurs.

The following information relating to each test suite parameter shall be provided in this section:

a) its name;

b) its type.

This information shall be provided in the format shown in Figure D-12/X.290, Part 2.

Test Suite Parameters		
Name	Type	Comments

.	.	.
.	.	.
.	.	.
(Name)	*(Type)*	*(Comments)*
.	.	.
.	.	.
.	.	.

FIGURE D-12/X.290, Part 2

Test suite parameters proforma

D.5.5 *Global constants*

The purpose of this section is to declare a set of names for values not derived from the PICS or PIXIT that will be constant throughout the test suite.

The following information relating to each global constant shall be provided in this section:

a) its name;

b) its type;

c) its value.

This information shall be provided in the format shown in Figure D-13/X.290, Part 2.

D.5.6 *Global variables*

A test suite may make use of a set of variables that are global to both the dynamic part and constraints part together. Typically, these variables will be used to reference values of PDU or ASP constraint fields, or components, from the dynamic part. Variables as in a conventional programming language (e.g. as counters).

All global variables to be used in a test suite shall be declared. The following information shall be provided for each variable declaration:

a) its name;

b) its type.

This information shall be provided in the format shown in Figure D-14/X.290, Part 2.

Global Constants			
Name	Type	Value	Comments

.	.	.	.
.	.	.	.
.	.	.	.
(Name)	*(Type)*	*(Value)*	*(Comments)*
.	.	.	.
.	.	.	.
.	.	.	.

FIGURE D-13/X.290, Part 2

Global constants proforma

Global Variable Declarations		
Variable Name	Type	Comments

.	.	.
.	.	.
.	.	.
(Variable Name)	*(Type)*	*(Comments)*
.	.	.
.	.	.
.	.	.

FIGURE D-14/X.290, Part 2

Global variable declarations proforma

Initially, all variables are unbound.

Variables may become bound (or be rebound) in the following contexts:

a) when the variable appears on the left-hand side of an assignment statements (D.6.7.1);

b) when the unbound variable appears in a Boolean expressions (§ D.6.7.1);

c) when the variable appears in a constraints reference (§ D.8).

D.5.7 *PCO declarations*

This section lists the set of points of control and observation (PCOs) to be used in the test suite and explains where in the testing environment these PCOs exist.

Note – The test method chosen determines the PCOs required to define the test suite.

The following information shall be provided for each PCO used in the test suite:

a) its name:

the name will be used in the Behaviour Section to specify where particular events occur;

b) an explanation of which type of tester is placed at the PCO and what role that tester plays.

This information shall be provided in the format shown in Figure D-15/X.290, Part 2.

PCO Declarations	
Name	Rôle
. . . *(Name)* . . .	. . . *(Rôle)* . . .

FIGURE D-15/X.290, Part 2

PCO declarations proforma

Example D-7 – An example PCO declaration is shown in Example D-8.

Example D-8 – PCO Declarations.

PCO Declarations	
Name	Rôle
L	SAP at the lower tester/(N-1)-service [lower tester is (N-1)-service user]
U	SAP at the upper tester/(N)-service [upper tester is (N)-service user]

Points of control and observation are usually just SAPs, but in general can be any appropriate points at which the test events can be controlled and observed. However, it is possible to define a PCO to correspond to a set of SAPs, provided all the SAPs comprising the PCO are:

a) at the same location (i.e., in the lower test or in the upper tester);

b) SAPs of the same service.

When a PCO corresponds to several SAPs the calling address (when initiating) or called address (when receiving) is used to identify the individual SAP.

Example D-9 – A typical example in which one PCO corresponds to several SAPs could be an Internet lower tester which uses one PCO representing all the *subnetwork points of attachment* for sending several Internet PDUs over different routes. Alternatively the same example may be written with several PCOs.

It is also possible to consider having a single SAP correspond to several PCOs. In this case there would be one PCO per connection.

Note – This makes it easier to identify each test event with the appropriate connection.

Finally it should be noted that a PCO may not be related to a SAP at all. For instance, this could be the case when a layer is composed of sublayers (e.g., in the Application layer, or in the lower layers, where a subnetwork point of attachment is not a SAP).

D.5.8 *ASP declarations*

This section lists the set of ASPs which may occur at the PCOs listed in § D.5.7.

Normally, the declared information can be found in the appropriate service definition. However, declaring it explicitly allows the addition of commentary specific to testing and to a particular test suite, as well as providing for cases where no explicit OSI* service definition exists (e.g. X.25).

The following information shall be supplied for each ASP:

a) its name;

if an abbreviated name is used, then the full name (as given in any appropriate service specification) shall follow in parentheses;

b) the PCO or PCOs at which it may occur;

all such PCOs shall have been declared in the PCO declarations section of the test suite;

c) whether or not a connection endpoint identifier is used to distinguish different instances of the ASP;

If it is stated that a connection endpoint identifier is used (see c), above), a parameter named ceid, of type *CEId* is available without further declaration,

upon receipt of an ASP which uses this parameter, the value of ceid shall become instantiated to the connection endpoint at which the ASP was received. This value is then available for use in the test case.

Example D-10 – Using a CEId:

PCO? AN-ASP [ceid = 3]

d) a list of the parameters associated with the ASP;

the following information shall be supplied for each parameter:

1) its name;

if an abbreviated name is used, then the full name (as given in any appropriate service specification) shall follow in parentheses;

2) its type.

ASP declarations have only one level of parameter. However, these parameters may be of an arbitrarily complex type (e.g. a complex ASN-1 type).

This information shall be provided in the format shown in Figure D-16/X.290, Part 2.

Example D-11 – Figure D-17/X.290, Part 2, shows an example from the Transport Service (Recommendation X.214). This could be part of the alphabet of ASPs used to describe the behaviour of an abstract upper tester in a DS test suite for the Class 0 Transport. CDA, CGA and QOS are user defined types. (See protocols, Recomendation X.224.)

ASP Declaration		
ASP: *(ASP)*	PCO: *(PCO) (list)*	CEId: Used or Unused

Service Control Information		
Parameter Name	Type	Comments
.	.	.
.	.	.
.	.	.
(Parameter Name)	*(Type)*	*(Comments)*
.	.	.
.	.	.
.	.	.

FIGURE D-16/X.290, Part 2

Abstract service primitive declaration

ASP Declaration		
ASP: CONreq (T-CONNECT request)	PCO: TSAP	CEId: USED

Service Control Information		
Parameter Name	Type	Comments
CdA (Called Address)	CDA	. . . of upper tester
CgA (Calling Address)	CGA	. . . of lower tester
QoS	Implementation dependent	Should ensure that Class 0 is used

FIGURE D-17/X.290, Part 2

Example abstract service primitive declaration

D.5.8.1 *ALP declarations*

ALPs are declared using a proforma similar to the ASP proforma. The following information shall be supplied for each ALP:

a) its name;

b) the PCO or PCOs at which it may occur;

c) a functional description of the ALP;

d) a list of the parameters associated with the ALP;

The following information shall be supplied for each parameter:

1) its name;

if an abbreviated name is used, then the full name (as given in any appropriate service specification) shall follow in parentheses;

2) its type.

This information shall be provided in the format shown in Figure D-18/X.290, Part 2.

ALP Declaration		
ALP: *(ALP)*	PCO: *(PCO list)*	Description: *(Functional Description)*

Service Control Information		
Parameter Name	Type	Comments
.	.	.
.	.	.
.	.	.
(Parameter Name)	*(Type)*	*(Comments)*
.	.	.
.	.	.
.	.	.

FIGURE D-18/X.290, Part 2

Abstract local primitive declaration

D.5.9 *Data type declarations*

The purpose of this section is to declare the data types which are used in the test suite. These types will be used mainly in ASP parameter declarations.

The most common data type declarations are PDU declarations. Other data type declarations may include ASN-1 type declarations, if appropriate.

D.5.9.1 *PDU declarations*

The declaration of PDUs is similar to that of ASPs. The following information shall be supplied for each PDU:

a) its name;

   if an abbreviated name is used, the full name (as given in any appropriate protocol specification) shall follow in parentheses;

b) a list of the parameters or, more generally fields associated with the PDU.

   Note – In order to be able to describe tests which exercise PDU encoding, it may be necessary to include fields (such as length indicators, for example) into the PDU description, even though they may not be considered to be PDU parameters in the protocol specification.

   The following information shall be supplied for each parameter:

   1) its name;

      if an abbreviated name is used, the full name (as given in any appropriate protocol specification) shall follow in parentheses;

   2) its type.

This information shall be provided in the format shown in Figure D-19/X.290, Part 2.

Data Type Declaration	
PDU: *(PDU)*	Comments: *[General Comments (e.g., restrictions on use)]*

Protocol Control Information		
Field Name	Type	Comments
.	.	.
.	.	.
.	.	.
(Field Name)	*(Type)*	*(Comments)*
.	.	.
.	.	.
.	.	.

FIGURE D-19/X.290, Part 2

Data type declaration proforma

Where more appropriate, a precise reference to an ASN-1 type description of a PDU can be used, instead of supplying all of the above information. In this case, the information shall be provided in the format show in Figure D-20/X.290, Part 2.

Data Type Declaration	
PDU Name	ASN.1 Type Definition

. . . *(PDU Name)* . . .	. . . *(ASN.1 Type Definition)* . . .

FIGURE D-20/X.290, Part 2

Data type declaration proforma

Example D-12 – Figure D-21/X.290, Part 2 shows an example ASN-1 PDU type definition from FTAM (ISO 8571).

ASN.1 Data Type Declaration	
PDU Name	ASN.1 Type Definition

F-INIT	ISO 8571-FTAM.PDU

FIGURE D-21/X.290, Part 2

ASN.1 Data type declarations

D.5.10 *Timers*

A test suite may make use of several types of timers. These types are used to distinguish the length of time the timers take to expire. A test case may make use of any number of instances of the same type of timer.

The following information shall be provided for each type of timer:

a) the timer type name;

b) the duration of the timer, which may be specified as a value or a range of values.

This information shall be provided in the format shown in Figure D-22/X.290, Part 2.

Timer Declarations		
Timer Type Name	Duration	Comments
.	.	.
.	.	.
.	.	.
(Timer Type Name)	*(Duration)*	*(Comments)*
.	.	.
.	.	.
.	.	.

FIGURE D-22/X.290, Part 2

Timer declarations proforma

Example D-13 – Figure D-23/X.290, Part 2 shows an example timer declaration.

Timer Declaration		
Timer Type Name	Duration	Comments
Receive-Timer	1. .3	Define a set of timers to be used when receiving data

FIGURE D-23/X.290, Part 2

Example timer declaration

D.5.11 *Abbreviations*

This section defines any abbreviations that are to be used in the rest of the test suite. Abbreviations are used as a macro facility, performing simple textual substitution operations. They may be used throughout a test suite.

An abbreviation may replace any piece of text within any single table box. The test case writer shall ensure that the resulting expansion follows the syntax of TTCN.

An abbreviation definition shall provide the following information.

a) an abbreviation identifier, or token;

b) its expansion, to be substituted for every occurence of the identifier throughout the test suite.

This information shall be provided in the format shown in Figure D-24/X.290, Part 2.

Abbreviations		
Abbreviation	Expansion	Comments
. . . *(Abbreviation)* . . .	. . . *(Expansion)* . . .	. . . *(Comments)* . . .

FIGURE D-24/X.290, Part 2

Abbreviations proforma

Example D-14 – Figure D-25/X.290, Part 2 shows an example abbreviation declaration from a Transport (Recommendation X.224) test suite.

Note – The ~ operator is defined in § D.6.8.

Abbreviation Declaration		
Abbreviation	Expansion	Comments
CR	N-DATAind [NSDU ~ CR-TPDU]	CR denotes any N-DATA indication whose Network Service Data Unit is the encoding of a Connection Request Transport Protocol Data Unit

FIGURE D-25/X.290, Part 2

Example abbreviations declaration

D.6 *Dynamic part*

The Dynamic Part contains the main body of the test suite – the test case and/or test step behaviour descriptions.

D.6.1 *Dynamic behaviour description*

The following information shall be supplied for the behaviour description of each test case, test step or set of related test steps:

a) a reference; the reference gives a name to the test case or step behaviour description. A test case reference shall conform to the requirements of § D.3.3.1. A test step reference shall conform to the requirements of § D.3.3.2;

b) an identifier; a reference reflects the structure of the test suite and the purpose of the test case or test step, and so may be quite lengthy. It is sometimes desireable to have a short name for a test case or test step. The identifier serves this purpose and may be used interchangeably with a reference. The test identifier shall be unique within a particular test suite;

c) a statement of purpose; this shall be an informal statement of purpose of the test case or test step or steps;

d) Defaults Reference; this shall be reference to a default behaviour description, if any, which applies to this behaviour specification (§ D.6.17);

e) a behaviour description; this section shall describe the behaviour of the tester or testers in terms of test events (and their parameters) in the tree notation described in § D.6.2.

This information shall be provided in the format shown in Figure D-26/X.290, Part 2.

Dynamic Behaviour

Reference: *(Test Case or Test Step Reference)*
Identifier: *(Identifier)*
Purpose: *(Purpose)*
Defaults Reference: *(Defaults Reference)*

Behaviour Description	Label	Constraints Reference	Verdict	Comments
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
(Behaviour Description)	*(Label)*	*(Constraints Reference)*	*(Verdict)*	*(Comments)*
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.

Synchronization Requirements *(optional)*

Extended Comments *(optional)*

FIGURE D-26/X.290, Part 2

Dynamic behaviour declaration proforma

D.6.2 *The tree notation*

The tree notation is used to provide the behaviour descriptions for a test suite. The behaviour descriptions are enumerations of possible observable sequences of test events.

Example D-15 – Suppose that the following sequences of events could occur during a test whose purpose was simply to establish a connection exchange some data, an close the connection:

a) CONreq, CONcnf, DATreq, DATind, DISreq

b) CONreq, CONcnf, DATreq, DATind, DISind

c) CONreq, CONcnf, DATreq, DISind

d) CONreq, CONcnf, DISind

e) CONreq, DISind

Progress can be thwarted at any time by the underlying service-provider. The tree notation simply factors out common initial sequences of the complete set. Using the tree notation, this would be written:

```
↑
A   Progression of Time →
l   EXAMPLE-TREE
t   CONreq
e     CONcnf
r       DATreq
n         DATind
a           DISreq
t           DISind
i         DISind
v       DISind
e     DISind
s
↓
```

Events at a same level of indentation represent the possible alternative events which may occur at that time. Alternative events shall be given in the order in which the tester shall attempt to complete them.

D.6.3 *Test events*

The names of events to be initiated by a tester shall be prefixed by an exclamation mark (!). Similarly, those which it is possible for a tester to accept shall be prefixed by a question mark (?).

Example D-16 – Using this convention, the Example Tree could be represented as follows:

```
EXAMPLE-TREE
!CONreq
   ?CONcnf
     !DATreq
       ?DATind
         !DISreq
         ?DISind
       ?DISind
     ?DISind
   ?DISind
```

Such names (composed of ! or ? followed by an event name) shall be prefixed by one of the PCO names appearing in the PCO list of the tree in which the event appears, unless the test suite only uses one PCO, in which case the PCO prefix may be omitted. The PCO name is used to indicate the PCO at which the test event may occur.

In cases where behaviour cannot be specified in terms of TTCN events (e.g. by virtue of the test method being used) then behaviour descriptions shall be given in natural language instead.

Example D-17 – Test Step from Generic Test Suite.

```
PARTIAL-TREE
?N-DATAind[UserData^DR]
 !N-DATAreq[UserData^DC]
    +POSTAMBLE

POSTAMBLE
/* close all open network connections */
```

D.6.3.1 *?OTHERWISE*

The predefined pseudo-event *(PCO)***?OTHERWISE** may be used to denote any ASP or ALP event which the tester may receive at the PCO.

Example D-18 – Assume that events A, B and C may be received at a given PCO:

```
?A [X=1]
?B
?OTHERWISE

 some behaviour ...
? TIMEOUT
```

is the same as

```
?A
?B
?A

 some behaviour ...
?B

 some behaviour ...
?C

 some behaviour ...
?TIMEOUT
```

When using ?OTHERWISE the following points should be noted:

a) ?OTHERWISE is always expanded with all possible events that may be received at the PCO since, even if an event is present at the same level, a Boolean expression or synchronisation requirement may prevent this event occuring;

b) if a tree makes use of multiple PCOs then a PCO shall be stated on the ?OTHERWISE statement;

c) ?OTHERWISE need not be the last among a set of alternatives;

d) ?OTHERWISE may have a Boolean expression and/or an assignment associated with it;

Example D-19 – Illustration of a qualified ?OTHERWISE

MAIN-TREE (X)

?A	pass
?B[X=2]	fail_1
?OTHERWISE[X=2]	fail_1
?C	fail_1
?OTHERWISE	pass

if the tree is invoked with X=2 then:
A => pass
B => fail_1
any other event => fail_1

if the tree in invoked with X ≠ 2 then:
A => pass
C => fail_1
any other event => pass

e) ?OTHERWISE does not prevent the use of default. The defaults are appended to the set of alternatives and will be examined in order. This means that ?OTHERWISE will invalidate some defaults but not necessarily all.

Example D-20 – Use of ?OTHERWISE in both the main and default tree:

MAIN-TREE
PCO1? A
PCO2? B
PCO1? OTHERWISE

DEFAULT-TREE
PCO2

?OTHERWISE	$fail_1$
PCO1? C	fail
?TIMEOUT	pass

The first and third events of the default are not invalidated by the ?OTHERWISE in the main tree.

D.6.4 *Tree names*

A behaviour description shall contain at least one behaviour tree. Each behaviour tree shall be prefixed by a tree name which is unique within a behaviour description (see § D.6.5).

Note – Many of the examples show tree names in bold text. This is for typographical clarity only and has no other significance.

D.6.4.1 *Formal PCOs*

Where a test suite involves the prescription of behaviour at more than PCO, a tree name shall be followed by a list of the (formal) PCO names used in the tree enclosed in brackets. Where a test suite only prescribes behaviour at one PCO, then this PCO list may be omitted.

Example D-21 – A tree involing the PCOs L and U might be named: **TREE-NAME[L,U]**.

D.6.4.2 *Formal parameters*

If a test step makes use of input parameters then after the PCO list (if any), a list of formal parameters to the tree may appear. These shall be enclosed in parentheses. TTCN does not support output parameters.

Example D-22 – **TREE-NAME[L,U](X,Y)**

D.6.5 *Tree attachment*

Trees may be attached to other trees by substituting the name of the tree to be attached, prefixed by a +, for an event name. The name of the attached tree shall be of the form:

a) <*TreeReference*>, or

b) <*TreeReference*> [<*ActualPCOs*>], or,

c) <*TreeReference*> (<*ActualParameters*>), or,

d) <*TreeReference*> [<*ActualPCOs*>](<*ActualParameters*>)

When a tree is attached to another tree, all the actuals are substituted for the formals using simple textual replacement. So an attached tree is analogous to a macro.

D.6.5.1 *Scope of attachment*

Behaviour descriptions may contain more than one tree. However, only the first tree in the behaviour description is accessible from outside the behaviour description. Any subsequent trees are considered to the test steps local to the behaviour description, and thus not externally accessible. Test cases, of course, are not attachable.

Thus the <*TreeReference*> shall be one of the forms:

a) <*TreeName*>

In this case <*TreeName*> shall be the name of one of the trees in the current behaviour description.

b) *<TestStepReference>/<TreeName>*

In this case, *<TreeName>* shall be the name of the *first* tree in the behaviour description of the test step referred to by the test step reference.

As usual, the equivalent *<TestStepIdentifier>* may be substituted for the *<test step reference>*.

Example D-23 – The following pair of trees:

```
MAIN-TREE [L,U] and                 SUB-TREE[X]
L!CONreq                             X?CONind
      + SUB-TREE[U]
```

is equivalent to:

```
COMPOUND-TREE [L,U]
L!CONreq
      U?CONind
```

Since tree attachment is really only a shorthand, variables are used in attached trees with the value they have at the point when the tree is attached, and may subsequently be used in assignment and/or Boolean expressions in the attached tree.

Attached trees may have parameters. Actuals are substituted for formals when the tree is attached.

Example D-24 – The following pair of trees.

```
MAIN-TREE        and                SUB-TREE (X,Y)
(M:= 1)                             (X:= Y)
    + SubTree(M,2)
        (M:= 3)
```

is equivalent to

```
COMPOUND-TREE
(M:= 1)
    (M:= 2)
        (M:= 3)
```

D.6.6 *Labels and GOTO*

A set of alternative events may be labelled by placing a lable in the lable column of the *first* event of the alternatives.

If a set of alternatives is labelled it is permitted to go to that set from any point in the tree. This is accomplished by placing → or the keyword GOTO followed by the name of a label that is defined in the same tree, immediately after an event. Should the event occur, the test proceeds with the set of alternatives referred to by the label.

Example D-25 – Figure D-27/X.290, Part 2 illustrates the use of the GOTO.

Events, not necessarily the first of a set of alternative events, may also have a label for the specification of synchronization (§ D.6.11) requirements. Such labels shall not be used as the object of a GOTO.

In the case where a first event needs to be labelled for both GOTO and for a synchronization requirement then a common label shall be used.

D.6.7 *Expressions*

The terms of an expression may be:

a) constants (including test suite parameters);

b) bound variables;

c) parameter values of the associated event (if any) referenced by the parameter name given in the event declaration;

d) references to encoded PDU field values in ASP parameters. These references take the form:

<ASP Parameter>.<FieldName>

Dynamic Behaviour				
Reference: Example/GOTO Identifier: GT Purpose: To illustrate the use of "GOTO" Defaults Reference: –				
Behaviour Description	**Label**	**Constraints Reference**	**Verdict**	**Comments**
/* Loop around echoing data. Pass, if data is received in response to each data sent.*/				
EchoUntilDis				
!DATAreq	Again			
?DATAind → again				Did not receive echo
?DISind			Fail	
?DISind			Pass	

FIGURE D-27/X.290, Part 2

Example illustrating the use of GOTO

D.6.7.1 *Assignment clauses*

Any event may be followed by a series of assignments. The effect of the assignment clause is to bind the global variable to the value of the expression if, and only if, the event occurs.

The assignments occur in the order in which they appear in the assignment clause. The expression shall contain no unbound variables.

D.6.7.2 *Boolean expressions*

An event may be qualified. This qualification is achieved by placing a bracketed Boolean expression after the event. This qualification shall be taken to mean the combination of the event occurring and the Boolean expression being satisfied.

If both a Boolean expression and an assignment clause are associated with the same event, the Boolean expression shall appear first.

The terms used in a Boolean expression can be any of the following:

a) constants (including test suite parameters);

b) variables;

if any unbound variables appear in a Boolean expression, then the event occurs only if values exist for the unbound variables which satisfy the Boolean expression. If the event does occur, all unbound variables become bound to any values consistent with their type and which satisfy the Boolean expression;

c) parameter values of the associated ASP (if any) referenced by the parameter name given in the ASP declaration;

d) references to a PDU encoding in the Constraints part (§ D.8). This encoding may be qualified by writing:

< PDUReference>[*< Boolean Expression>*]

where the terms of the Boolean Expression may be references to fields of the referrenced PDU;

e) references to encoded PDU field values in ASP parameters in the Constraints Part. These references take the form:

< ASPParameter>.*< FieldName>*

D.6.7.3 *Assignment clauses and Boolean expressions with events*

It is allowed to associate an event with either an assignment, or a Boolean expression or both. However, only the following combinations may be used:

a) *<Event>* The event is not qualified.

b) *<Event>* (*<Assignment>*) The assignment is executed only if the event occurs.

c) *<Event>*[*<Boolean Expression>*] The event may occur only if the boolean expression holds.

d) *<Event>*[*<Boolean Expression>*](*<Assignment>*) The assignment is executed if, and only if, both

   1) the event occurs, and

   2) the Boolean expression holds.

   Since all variables ocurring in the Boolean expression become bound, they may be used in the expression which is part of the assignment.

Boolean expressions may be broken down into an uninterrupted set of Boolean expressions. This facility may be useful when used in conjunction with abbreviations (§ D.5.11). Similarly, assignments may be broken down into an uninterruped set of assignments.

Example D-26 – PCO? CR[X=1][Y < 3] *is equivalent to* PCO? CR[(X=1)AND(Y < 3)]

D.6.7.4 *Assignment clauses and Boolean expressions without events*

It is permitted to use assignment clauses and Boolean expressions by themselves, without any associated event.

Only the following compositions are allowed:

a) (*<Assignment>*) The assignment is executed only if the event occurs.

b) [*<Boolean Expression>*] The event may occur only if the Boolean expression holds.

c) [*<Boolean Expression>*](*<Assignment>*) the assignment is executed only if the Boolean expression holds.

D.6.8 *The encode/decode operator*

The encode/decode operator allows the specification of the encoding of PDU fields. The syntax for this is defined in § D.6.8.2. The ~ operator should be read as "is the encoding of".

Example D-27 – Consider the event:

N-SAP? N-DATAind[UserData ~ CR-TPDU]   CR1

means that the event matches if, and only if, at N-SAP we receive an N-DATAind whose UserData field is the correct encoding of a CR-TPDU according to the constraint CR1 (see § D.6.13).

Suppose that F1 is a field of the constraint CR1. If this event occurs, then F1 can be referenced by writing UserData.F1.

Note – If the content of F1 is a free variable (§ D.8.2) or a global variable (§ D.5.6), then the event occurring will bind this variable (See § D.6.7.2).

Similarly:

N-SAP! N-DATAreq[UserData ~ CC-TPDU]   CC1

means that the tester sends an N-DATAreq whose UserData field is the encoding of a CC-TPDU according to the constraint CC1.

D.6.8.1 *Aliasing*

A simple mechanism for preserving the values of entire PDUs is provided. The alias is an implicitly bound variable that may be used as a shorthand or to distinguish between two fields (PDU or ASP) that have the same name.

Example D-28 – Use of the alias:

N-SAP? N-DATAind[UserData\UD1 ~ CR-TPDU[UserData\UD2 ~ TM-PDU]]   CR1,TM1

the aliases UD1 and UD2 are bound to their respective UserData [or, more precisely, bound to the constraints CR(CR1) and TM(TM1)]. Suppose that F1 is a field of CR-TPDU(CR1) and that F2 is a field of the constraint TM(TM1). Then these fields may be referenced as UD1.F1 and UD2.F2.

D.6.9 *Parallel trees*

It is sometimes convenient to be able to describe behaviour in terms of separate behaviour descriptions occurring in parallel. Typically, each separate description will refer to behaviour at a different PCO.

Parallel trees are denoted as follows:

$$|| \, \textbf{Tree}_1 \, , \ldots \, , \textbf{Tree}_n$$

The following rules shall apply:

a) the subscript n shall be greater than 1;

b) the tree which splits its behaviour into parallel trees shall have more than one PCO and at least as many PCOs as there are parallel trees;

c) all PCOs of the current tree shall be a PCO of one and exactly one of the parallel trees;

d) there shall be no timer(s) running or suspended;

e) there shall be no other event at the same level of alternative;

f) only the first tree in the statement (i.e. Tree$_1$) may assign a verdict;

g) when the first tree (i.e. Tree$_1$) terminates:

   1) If Tree$_1$ terminates with a verdict assigned then the test case terminates with that verdict;

   2) If Tree$_1$ terminates without a verdict (i.e. a tip of Tree$_1$ is reached) then the behaviour continues as if Tree$_1$ was attached.

      A) Tree$_2$ ... Tree$_n$ are disregarded;

      B) subsequent behaviours of the main tree are appended to the tips of Tree$_1$;

   3) if a tip of Tree$_i$ ($i > 1$) is reached, then:

      A) if any further event occurs at a PCO of Tree$_1$ then this is an error;

      B) otherwise (g)2 will apply (i.e. Tree$_1$ has to terminate at some point).

Example D-29 – Lower and upper tester behaviour as parallel trees:

```
MAIN-TREE [UT,LT]
  +connect-link-layer(LT)
    UT! N-CONreq
      || LT-TREE(LT),UT-TREE(UT)
        +disconnect-link-layer(LT)                    pass

UT-TREE [UT]
  UT? N-CONcnf                             L
    UT! N-DTAreq → L
    ? Otherwise → L
  UT? Otherwise → L

LT-TREE [LT]
  LT? N-DTAind [call-request]
    LT! N-DTAreq [call-accept]
      LT? B-DTAind [dt-packet]

DEFAULT-TREE
UT? Otherwise
  +disconnect-link-layer                              fail1
```

D.6.10 *Timer management*

It is assumed that the timers used in a test suite will always be in one of the following states:

a) inactive,

b) running,

c) suspended.

D.6.10.1 *Timer operations*

A set of "operations", which may be used like assignments, are used to model timer management. These operations can be applied to:

a) a set of timers;

   This is specified by following the timer operation with just the timer type name.

b) an individual timer from a given set of timers of the same timer type.

This is specified by following the timer operation by the timer type name and an identifier for the individual timer. A timer identifier is a variable name.

Note – It is not necessary to declare timer identifiers explicitly. They are declared implicitly when stated as a parameter on the **Start** operation.

D.6.10.2 *Definition of timer operations*

The timer operations defined are:

a) **Start** < *timer type*> [, < *timer id*> [, <value>]]

The **Start** operation is used to indicate that an inactive timer, or set of timers, should start running. (If the timer identifier parameter is omitted, this shall be interpreted as starting a timer of the given timer type. However, it is not then possible to manipulate this timer separately from any other timers of this type which may be defined.)

The optional value parameter shall be used if it is desired to assign an expiry time (i.e., duration) for a timer. This value is not required to be within the range of values defined for the timer. This provides the ability to test timer duration situations. Otherwise, any time within the range of values specified in the Declarations Part may be used.

b) **Cancel** < *timer type*> [, < *timer id*>]

The **Cancel** operation is used to indicate that a running (or suspended) timer is to become inactive. (If the timer identifier parameter is omitted, all suspended timers of the given timer type shall become inactive.)

c) **Suspend** < *timer type*>[, < *timer id*>]

The **Suspend** operation is used to indicate that a running timer is to become suspended. (If the timer identifier parameter is omitted, all timers of the givent type shall become suspended.)

d) **Resume** < *timer type*> [, < *timer id*>]

The **Resume** operation is used to indicate that a suspended timer is to become (i.e. resume) running. (If the timer identifier parameter is omitted, all timers of the given timer type shall resume running.)

D.6.10.3 *Timer pseudo-event*

In addition to the timer operations, two timer pseudo-events are defined.

a) The **timeout** pseudo-event, which has the following form:

?Timeout < *timer type*> [, < *timer id*>]

This pseudo-event may be used within a behaviour tree to check for expiration of the specified timer. The timer identifier parameter may be omitted if:

1) there is only one timer of type timer type defined;

2) there are multiple timers of type timer identifier defined, and it is not necessary to distinguish between them.

D.6.10.4 *The elapse pseudo-event*

An **elapse** statement takes the following form:

Elapse < *timer type*> [, < *value*>]

An **elapse** statement puts an upper bound on the time in which a tree will remain at a given level of indentation (i.e. in a given state). When using **elapse** the following points should be noted:

a) the **elapse** statement shall not have any Boolean expression or synchronization requirements associated with it;

b) if more than one **elapse** is used then only the one which specifies the smallest time duration will be considered;

c) it is recommended (but not mandatory) that the **elapse**, if any, be the last of a set of alternatives;

d) the alternative specified by the **elapse** occurs only if, during the specified duration, no:

1) other alternative is scheduled;

2) error occurs (i.e. an even for which there is no alternative).

Note – Although elapse uses a timer-type for the convenience of giving a value and units, the elapse does not imply that any user accessible timer is started.

Example D-30 – Using **Elapse**:

?A

?B

ELAPSE

means that the tester will wait on the events ?A or ?B only until one of them occurs or until the ELAPSE expires.

D.6.11 *Synchronization*

The ability to specify the relative ordering of events in different trees, which may or may not be in the same behaviour description, is provided by the synchronization statement. The synchronization statement is a Boolean expression of the form:

a) $(\mathit{TreeReference}_1)/(\mathit{Label}_1) < (\mathit{TreeReference}_2)/(\mathit{Label}_2)$OR

b) $(\mathit{TreeReference}_1)/(\mathit{Label}_1)_1 > (\mathit{TreeReference}_2)/(\mathit{Label}_2)$

where the predicates are to be interpreted as:

a) the event labelled by (*Label*$_1$) in test step (*TreeReference*$_1$) *occurs* **before** the event labelled (*Label*$_2$) in test step (*TreeReference*$_2$);

b) the event labelled by (*Label*$_1$) in test step (*TreeReference*$_1$) *occurs* **after** the event labelled (*Label*$_2$) in test step (*TreeReference*$_2$).

Note 1 – Synchronization is specified in terms of events in different trees rather than in terms of choices in the same tree.

Note 2 – Synchronization may only be used between parallel trees, which may, or may not, be part of the same behaviour description.

Note 3 – An alternative choice to synchronized parallel trees is one single tree dealing with several PCOs.

D.6.11.1 *Interpreting synchronization statements*

A synchronization statement shall be interpreted as follows:

IF

a) an event among a set of alternatives has a label L AND,

b) there exists a synchronization statement containing (*TreeReference/Label*) in its right or left part, AND

c) (*TreeReference*) designates the behaviour description in which the event is

THEN

to the Boolean expression used to decide whether the event is selected append the following:

"AND the event designated by the label in the other part of the synchronization statement already occurred (respectively did not already occur)".

Therefore a synchronization statement does not mean that the tree is "suspended" until the synchronization requirement is satisfied, but rather that the event involved in the synchronization is not a candidate for selection among the current set of alternatives if the synchronization is not satisfied. The normal rules for selecting an event apply, considering the synchronization as an extra Boolean expression. If no event can be selected then an error may occur if an event is present at the PCO which cannot be flow-controlled. The test writer shall therefore include the appropriate event at the same level of indentation.

Example D-31:

```
FIRST-TREE
LT! L-DTAreq [DT packet with D-bit set]            L1
      LT? L-DTAind [P(R) for this packet]          L2       fail$_1$
      LT? L-DTAind [P(R) < this packet] → L2
      LT? L-DTAind [P(R) < this packet]            L3       pass
SECOND-TREE
UT? N-DTAind                                       L
```

with the synchronization requirements:

FIRST-TREE/L3 > SECOND-TREE/L
FIRST-TREE/L2 < SECOND-TREE/L

D.6.12 *The repeat construct*

This section describes a mechanism in TTCN behaviour descriptions to iterate a test step a variable number of times. The form of the construct is:

REPEAT (*TreeReference*) [(*Varid*)]

and it must be followed by a set of alternative Boolean expressions (guards). It should be noted that:

a) the guards need not be mutually exclusive. The repeat terminates when at least one guard holds;

b) no other event or pseudo-event may appear at the same level of indentation as the guards;

c) the VARid (when specified)

  1) may be used within the repeated tree;

  2) may be used inside the guards;

  3) is not a global variable (i.e. its scope is local to the REPEAT);

  4) is implicitly of type Integer.

D.6.12.1 *Repeat with no VARid*

When VARid is not specified then the construct:

```
REPEAT TreeReference ($a_1$, ... , $a_n$)
    [$b_1$]
        ...
    .
    .
    .
    [$b_n$]
        ...
```

shall be interpreted as meaning:

```
+TreeReference ($a_1$, ... , $a_n$)L
    [$b_1$]
        ...
    .
    .
    .
    [$b_n$]
        ...
    [true] → L
```

Where (a_1, ..., a_n) are the actual tree parameters (if any).

D.6.12.2 *REPEAT with VARid*

When VARid is specified the construct:

```
REPEAT TreeReference ($a_1$, ... , $a_n$) [VARid]
```

expands to:

```
+ repeat-tree ($a_1$, ... , $a_n$, 0)
```

and this repeat-tree is defined as:

```
Dummy-repeat-tree ($f_1$, ... , $f_n$, VARid)
    + treeReference ($f_1$, ... , $f_n$, VARid)      L
    (VARid:=VARid+1)
    [$b_1$]
        ...
    .
    .
    .
    [$b_n$]
        ...
     [true] → L
```

Where (a_1, ... , a_n) are the actual tree parameters (if any) and (f_1, ... , f_n) are the formal tree parameters and tree-id (f_1, ... , f_n) is redefined as tree-id (f_1, ... , f_n, VARid)

D.6.13 *Constraints reference*

This column allows reference to be made to an ASP, ALP, or PDU that is defined in the Constraints declaration part (§ D.8 – which also provides more information on encoding constraints).

Example D-32 – A constraint reference:

N-SAP? N-DATAind [UserData ~ CR-TPDU]   N-DATAind (D1), CR-TPRU (CR1)

Where N-DATAind and CR-TPDU are defined in the ASP and Data Type Declarations sections of the test suite. The constraints D1 and CR1 are defined in the constraints section of the test suite.

If an event is qualified by a Boolean expression and also has a Constraints Reference, this shall be interpreted as *the event occurs if, and only if, both the Boolean expression and the constraint hold.*

If an event is followed by an assignment clause and also has either or both a Constraints Reference or a Boolean expression, this shall be interpreted as *the assignment is performed if, and only if, the event occurs,* according to the definition given above.

D.6.14 *Verdicts*

A test clause shall end with a verdict. This verdict is given with respect to the test purpose, and may be one of:

a) Pass;

b) Fail_1;

c) Fail_2 (test purpose accomplished but failed subsequently)

d) Fail_3 (protocol error but test purpose inconclusive);

e) Inconc(lusive).

A test event may have a verdict assigned to it by using the verdict column. Entries in the verdict column will either contain one of the following or be left blank (i.e. no verdict). This particular case is referred to in the following as the special value *none.*

D.6.14.1 *Rules for applying verdicts*

The following rules for verdict assignment apply:

a) at any time during the execution of a test case, there is a "default verdict";

Note – Not to be confused with any verdict that may appear in a default behaviour description.

b) at the beginning of a test case the default verdict is *none*;

c) when a tree is attached the default verdict for the attached tree becomes the result of applying Figure D-28/X.290, Part 2. "Current" is the current default verdict and "New" is the verdict assigned at the point of attachment.

d) when an event (other than attach) has a verdict assigned to it which is not "none" (i.e. the verdict column is not blank) the test case ends with the verdict being the result of applying Figure - D-28/X.290, Part 2 to the default verdict and the verdict assigned to that event;

e) when the tip of a tree that is not the main tree is reached the test case terminates with the verdict being the default verdict.

Note – In this case there is no explicit verdict assigned to the last event of the tree (otherwise § D.6.14.1 d) would apply). This is an error.

Example D-33 – The use of a verdict attached to a subtree:

```
A-TREE
!CONreq
?CONcnf
    + Data Transfer ("Hello")
        [ok = "yes"]
            + bye  Pass
        [ok = "no"]
            + bye  Fail
?DISind                    Inconc
```

	New					
Current	none	Pass	Fail$_1$	Fail$_2$	Fail$_3$	Inconc
none	none	Pass	Fail$_1$	Fail$_2$	Fail$_3$	Inconc
Pass	Pass	Pass	Fail$_2$	–	–	–
Fail$_1$	Fail$_1$	–	Fail$_1$	–	–	–
Fail$_2$	Fail$_2$	–	Fail$_2$	Fail$_2$	–	–
Fail$_3$	Fail$_3$	–	Fail$_1$	–	Fail$_3$	–
Inconc	Inconc	–	Fail$_3$	–	–	Inconc

FIGURE D-28/X.290, Part 2

Default Verdict Precedence

D.6.15 *Comments*

This column contains short remarks, or references to extended comments given at the bottom of the table.

D.6.16 *Defaults reference*

In order to emphasize the main path through a test, it is possible to specify, separately from the main behaviour description, default subsequent behaviours for any event which a tester may receive. The main behaviour descriptions of the dynamic part are completed by appending every set of alternatives with subsequent behaviour from a default behaviour description.

In order to ensure that a test case specification is complete, a TTCN test specification shall specify subsequent behaviour for every event possible, either in a main behaviour description, or by means of an associated default behaviour description.

Note – This can be simply done with ?OTHERWISE.

Default behaviours only apply where the relevant test event is possible.

Example D-34 – The following test case could be split into two behaviour descriptions as:

```
EXAMPLE-TREE
   !CONreq
      ?CONcnf
         !DATreq
            ?DATind
               !DISreq
```

with a trivial default (having no subsequent behaviour associated with it) of:

```
DEFAULT-TREE
?DISind
```

The Defaults Reference is used to associate a set of defaults behaviours with a main behaviour description.

D.6.17 *The specification of defaults*

The default behaviour descriptions shall be presented in tables in the Default section of the Dynamic Part of a test suite. These tables shall be of the format shown in Figure D-29/X.290, Part 2.

<table>
<tr><td colspan="5">Default Dynamic Behaviour</td></tr>
<tr><td colspan="5">Reference: <*Test Case or Test Step Reference*>
Identifier: <*Identifier*>
Purpose: <*Purpose*>
Defaults Reference: <*Defaults Reference*></td></tr>
<tr><td>Behaviour Description</td><td>Label</td><td>Constraints Reference</td><td>Verdict</td><td>Comments</td></tr>
<tr><td><*Behaviour Description*></td><td>.
.
.
<*Label*>
.
.
.</td><td>.
.
.
<*Constraints Reference*>
.
.
.</td><td>.
.
.
<*Verdict*>
.
.
.</td><td>.
.
.
<*Comments*>
.
.
.</td></tr>
<tr><td colspan="5">Extended Comments *(optional)*</td></tr>
</table>

FIGURE D-29/X.290, Part 2

Dynamic Behaviour Declaration Proforma

These tables shall be entirely analogous to the main behaviour descriptions except for the following differences:

a) the default behaviour tree shall not have a name;

b) it shall contain a single unnamed tree; (*/Rightarrow* PCOs and parameters cannot be passed to defaults);

c) it may not contain a Synchronization Requirements entry.

Note – References to default dynamic behaviours obey the same rules as for test steps (§ D.3.3.2).

Example D-35 – Figures D-30/X.290, Part 2, D-31/X.290, Part 2, and D-32/X.290, Part 2, present a slightly augmented version of EXAMPLE-TREE. The default behaviours in Figures D-31/X.290, Part 2 and D-32/X.290, Part 2, in combination, are exactly equivalent to the behaviour shown in Figure D-33/X.290, Part 2.

Dynamic Behaviour

Reference: Examples/BiggerExample Tree
Identifier: BET
Purpose: To illustrate the use of defaults
Defaults Reference: Examples/Defaults/Standard

Behaviour Description	Label	Contraints Reference	Verdict	Comments
BIGGER-EXAMPLE-TREE				
!CONreq				Request . . .
?CONcnf				. . . Confirm
!DATreq				Send Data
!DATind				Accept Data
!DISreq			Pass	Disconnect

FIGURE D-30/X.290, Part 2

Default Behaviour Example (Part 1 of 3)

Default Dynamic Behaviour

Reference: Examples/Defaults/Standard
Identifier: Std
Purpose: First Level of Defaults
Defaults Reference: Examples/Defaults/Default

Behaviour Description	Label	Constraints Reference	Verdict	Comments
?RSTind				Reset
!DISreq			Inconc	Finish

FIGURE D-31/X.290, Part 2

Default Behaviour Example (Part 2 of 3)

Default Dynamic Behaviour

Reference: Examples/Defaults/Default
Identifier: Def
Purpose: Second Level of Defaults
Defaults Reference: None.

Behaviour Description	Label	Constraints Reference	Verdict	Comments
?DISind			Inconc	Premature

FIGURE D-32/X.290, Part 2

Default Behaviour Example (Part 3 of 3)

Default Dynamic Behaviour

Reference: Examples/Defaults/Standard
Identifier: Std
Purpose: Combined Defaults
Defaults Reference: None

Behaviour Description	Label	Constraints Reference	Verdict	Comments
?RSTind				Reset
!DISreq			Inconc	Finish
?DISind			Inconc	Finish
?DISind			Inconc	Premature

FIGURE D-33/X.290, Part 2

Default Behaviour Example (Parts 2 and 3 combined)

D.7 *Interpreting trees*

This section describes how to interprete TTCN behaviour descriptions and how to assign a verdict to a test case. It consists of two parts intended for different audiences:

a) the first part provides a set of rules together with a simple algorithm intended to aid the understanding of TTCN;

b) the second part provides a more comprehensive algorithm (the rules have been integrated into the algorithm) intended to help implementing TTCN.

D.7.1 *Interpretation of TTCN (partial algorithm)*

a) a TTCN tree has a number of levels of event hierarchy that is greater than or equal to one. Execution of the events in the tree will terminate at some level by:

   1) EITHER a verdict being assigned;

   2) OR the occurrence of a test case error (i.e. a tip with no verdict is reached or an unexpected event is received at a PCO);

b) each level has a number of alternative events greater or equal to one;

c) we assume here that:

   1) the tree expansion has already been done according to the general rules given in § D.6.5;

   2) the set of defaults to be applied has been computed according to the rules given in § D.6.14.1;

   3) the rules for default verdicts in the case of a subtree are given in § D.6.14.1;

d) the algorithm to interpret what to do at a given level is as follows:

   1) append the defaults to the set of alternatives;

   2) use the procedure in § D.7.3 to determine whether a match occurs;

   3) if the procedure in § D.7.3 returns *no match* then TERMINATE (error);

   4) if a match is found then:

      A) if the event which causes the match has a verdict assigned to it, or if it is the tip of a subtree with a default verdict, then terminate the test case and return the final verdict determined according to Figure D-29/X.290, Part 2;

      B) otherwise determine the next level of indentation as the one following the event which caused the match and:

         i) if that level designates a non-empty set of alternatives, then restart processing for that level;

         ii) otherwise TERMINATE (error).

D.7.2 *Rules for default*

a) Defaults of a subtree take precedence over (i.e. are to be considered before) those defaults of the tree which attached the subtree.

   Example D-36 – Defaults in both main and attached tree:

   MAIN-TREE
   ?A
       + Step-1
   DEFAULT-TREE
   ?B   fail$_1$
   ?E   fail$_1$
   Step-1
   ?C
       ?D
   Default-Step-1
   ?B   pass
   and the sequences:
   ?A ?C ?B => pass
   ?A ?C ?E => fail$_1$

b) defaults of a subtree are considered only when the subtree is actually entered (i.e. at a level of indentation greater than those of the events causing entry into that subtree);

c) defaults cease to apply at the end of a tree and are not to be appended to the "empty alternative set" which follows a tip of the tree. This is true for the main tree as well as the subtree.

Example D-37:

```
MAIN-TREE
?A
    ?B
    ?C                          fail_1
?OTHERWISE                      pass

DEFAULT-TREE
?OTHERWISE                      inconc
```

and the sequence ?A ?B ?ANYTHING is not valid (i.e. the test case is incomplete);

d) when defaults are used in a subtree and an event occurs which matches the default, if the sequence specified by the default does not have a verdict, the execution will continue in the main tree after a tip of the default is reached, except, of course, if the subtree has a default verdict!

Example D-38 – Default tree with verdicts:

```
MAIN-TREE
?A
    + Step-1
      ?X                        pass
?B                              fail_1

Step-1
?D
    ?E

Default
?B
    !C
```

and the sequence ?A ?D ?B ?C ?X => pass

D.7.3 *Matching events*

The rules for determining if an event among a set of alternatives matches (i.e. evaluates as having occurred) are listed below for each possible type of TTCN event or pseudo-event:

a) **!(ASPid | ALPid | PDUid)** – for the SEND event to match, it must be possible to send the ASP, ALP or PDU (e.g. considering flow control). If a PCO has been specified this determination is made with consideration to that particular PCO. Additionally all qualifying Boolean expressions and/or constraints placed on the ASP, ALSP or PDU must hold;

b) **?(ASPid | ALPid | PDUid** – for the RECEIVE event to match the specified ASP, ALP or PDU must have been received and all specified constraint must hold. If a PCO has been specified the ASP, ALP or PDU must have been received at that particular PCO. Additionally all qualifying Boolean expressions and/or constraints placed on the ASP, ALSP or PDU must hold;

c) **?TIMEOUT** – for the Timeout pseudo-event to match, a timer of the named type must have expired. If an optional timer identifier is specified, then the expiration of only that individual timer is considered;

d) **ELAPSE** – for the Elapse event to match, the timer which was implicitly started (by the specification of the Elapse) must have expired before any of the events prior to the Elapse have matched;

e) **?OTHERWISE** – for the Otherwise event to match, some event must have been received which has not matched any previous alternatives to the Otherwise. If a PCO is specified then the event must have been received at that particular PCO;

f) **ATTACH** – this event will never be considered for matching because all tree attachments are assumed to have been fully expanded before attempting to match an alternative;

g) **REPEAT** – this event will never be considered for matching because it will have previously been expanded;

h) **GOTO** – this event always evaluates to TRUE, and therefore always matches;

i) **BOOLEAN EXPRESSIONS** – a Boolean expression stated as an alternative matches if the expression evaluates to TRUE (see § D.6.7.2);

j) **ASSIGNMENT CLAUSES** – an assignment clause always matched;

k) **PARALLEL TREES** – the execution of parallel trees will never be considered for matching because these will have been fully expanded before attempting to match an alternative;

l) **TIMER OPS Start, Cancel, Suspend** and **Resume** – all timer operations will always match.

Note – Any events specified following and at the same level of indentation as a goto, assignment clause or a timer operation can never be reached.

D.7.4 *Comprehensive algorithm*

For the purposes of this section:

a) a verdict may be one of the following:

1) pass, fail$_1$, fail$_2$, fail$_3$, inconc;

2) none, which is used:

A) as a special value returned when a tip of a sub-tree is reached without a verdict;

B) as a default verdict value when attaching a sub-tree for which there is no default value;

C) as the verdict assigned to an event when the verdict column is empty;

3) an ordered union of an ordered set is . . . (append?);

b) let EVALUATE-DEFAULT (*R*) be a function:

1) of a behaviour reference (*R*);

2) which returns an ordered set of alternative events;

3) this ordered set is defined as follows:

A) let *d* be the reference of the default behaviour associated with the behaviour description referenced to by *R*;

B) if there is no *d* return the empty set;

C) otherwise return the ordered union of:

i) all the events at the first level of indentation in the behaviour description referenced to by *d*;

ii) and evaluate EVALUATE-DEFAULT (*d*).

c) let SELECT (*E*) be a function:

1) of an ordered set *E* of alternative events:

2) which returns an event of its input set, or the special value error;

3) which is defined as follows:

A) let *E'* be a set of 2-uple {event, event} initialized as:

$\{ \{ e_1, e_1 \}, \{ e_2, e_2 \} \dots \{ e_n, e_n \} \}$

where $e_1 \dots e_n$ are the elements of E;

B) replace any element of *E'* of the form:

$\{ +\text{tree}, e \}, \dots \forall e$ by $\{ t_1, e \} \dots \{ t_n, e \}$

where $t_1, \dots t_n$ are the events at the first level of indentation of the tree;

Note 1 – This has to be done (possibly recursively) until no element is of the form $\{ +\text{tree}, e \}$ since a tree may attach a subtree at its first level of indentation.

Note 2 – The second element of the 2-uple keeps track of the original event name in the ??? of ???, SELECT will return this element.

Note 3 – *E'* is an ordered set.

C) if there is any **ELAPSE** among the alternatives then compute T = time-unit (elapse value) otherwise set T = infinity;

D) for each element in { *x*, *e* } in *E′* do the following:

IF the following holds:

i) IF there is a Boolean expression and it holds

AND

- EITHER x is PCO? Otherwise and any ASP event is ready at that PCO;
- OR x is PCO? ASP and the ASP is ready and any Boolean expression using parameters of this ASP holds and all the constraints (if any) are satisfied;
- OR x is PCO! ASP and ASP can be sent at this PCO (i.e. flow control permits sending) and all constraints (if any) are satisfied;
- OR x is neither ? nor !.

THEN SELECT terminates and returns the event.

E) if there is any ASP ready at any PCO and this ASP cannot be flow-controlled, or there is an elapsed timer, SELECT terminates and returns "error";

F) otherwise

i) if a period of time more than T has elapsed since c(3)D was entered for the first time, then select the event "elapse" which was used to compute T;

ii) otherwise restart in c(3)D;

i.e. wait for the next event (busy loop);

d) let EXECUTE(x) be a function

1) of an event x

2) which terminates without returning a value

3) is defined as follows:

A) if x is ? OTHERWISE, then discard the actual event (?ASP) which caused the SELECT function to return, and then;

B) if x is PCO? ASP, take the ASP from the PCO, bind any related parameters or free variables to the received values; and then;

C) bind any unbound global variables so that the Boolean expression holds (if any); and then;

D) if x is PCO! ASP, bind the parameters to values so that the Boolean expression holds and issue the ASP at the PCO; and then;

E) perform any assignments (including any timer operation);

e) let EVAL-TREE be a function:

1) of the following:

- the current level of indentation (L) which designates a set of alternative events in a behaviour description;
- the current default verdict (V) which indicates in the case of tree attachment whether a verdict was assigned at the point of attachment of the tree.

  Note – none means no verdict.
- the ordered set of default event (*D*) to be used to interpret the tree;

2) EVAL-TREE returns a value as defined in a;

3) EVAL-TREE is defined as:

A) evaluate *K* = set of events as L;

B) invokes SELECT(*K* ∪ *D*);

C) if e(3)B returns error, then EVAL-TREE TERMINATES(error);

D) if e(3)B returns an event of the form + *tree* then:

i) compute V = EVAL-TREE (*L′*, *V′*, *D′*) where

- *L′* designates the first level of indentation in the attached tree
- *V′* is the result of applying Figure D-29/X.290, Part 2 and the verdict assigned to the tree
- *D′* is the ordered union of EVAL-DEFAULT (tree-ref) and D

ii) if V ≠ none, then return V

E) otherwise:

- invoke EXCUTE(e) where e is the event as returned by e(3)B;
- if e has verdict compute V as the result of applying Figure D-29/X.290, Part 2 and the verdict assigned to the event, return V;

F) compute L = next level of indentation, taking into account any **GOTO**;

G) if L designates an empty set then terminte and return V, otherwise continue with e(3)A.

4) the interpretation of a test case is given by EVAL-TREE (L, none, D) where:

L = the first level of indentation of the first tree in the behaviour description of the test case;

D = EVALUATE-DEFAULT (test-case-reference);

A) if EVAL-TREE returns "error" then the specification is incomplete;

B) if result is "none" then the test case specification is incomplete;

C) otherwise EVAL-TREE returns the verdict.

D.8 *Constraints declaration part*

It is necessary to describe, in detail, the coding of parameters in PDUs and ASPs. The parameter coding is described using either a tabular method (§ D.8.1.1), or a method which takes advantage of the ASN-1 notation (§ D.8.3). Reference to particular values is made in the Constraints Reference column of the tables used in the Dynamic Part (§ D.6.13).

D.8.1 *Parameter encodings*

An ASP or PDU can be considered as a list of parameters. However, because of the importance of PDU encoding in testing, it may be more appropriate not to abstract away from PDU encoding and to consider a PDU as a list of fields such as Length, Parameter Type, Parameter Value.

D.8.1.1 *Tabular method*

If an actual value is assigned to each field, the actual PDU or ASP can be represented by a list of values. Secondly, a set of lists can be built, each element of this set being a possible list of values from all the combinations. Then, for each PDU or ASP, this set can be represented by a table. The PDU constraints declaration table shall have the proforma shown in Figure D-34/X.290, Part 2.

<table>
<tr><td colspan="3">PDU Constraint</td></tr>
<tr><td>PDU Name: <*PDU Name*></td><td colspan="2">Constraint Name: <*Constraint Name*></td></tr>
<tr><td>Field Name</td><td>Value</td><td>Comments</td></tr>
<tr><td>.
.
.
<*Field Name*>
.
.
.</td><td>.
.
.
<*Value*>
.
.
.</td><td>.
.
.
<*Comment*>
.
.
.</td></tr>
</table>

FIGURE D-34/X.290, Part 2

PDU Constraint Proforma

Note – In the following text there is a corresponding ASP or ALP table for every occurrence of a PDU table. The ASP or ALP proforma is similar in every detail except that every occurrence of PDU in the table fields substitute the word ASP or ALP.

Each field entry in the field name column shall have been declared in the PDU (or ASP, ALP) declaration. Values assigned to each field shall be of the type specified in the PDU (or ASP, ALP) declaration.

In cases where a constraint contains only a few fields, or when there are only a small number of constraints, the compacted version of the constraints table proforma shown in Figure D-35/X.290, Part 2 may be used.

PDU Constraints				
PDU Name: *<Name>*				
Constraint Name	Field Name			Comments
	<Field Name>$_1$	. . .	*<Field Name>*$_n$	
<Constraint Name>$_1$	*<Value>*$_{1.1}$	. . .	*<Value>*$_{1.n}$	*<Comment>*$_1$
<Constraint Name>$_2$	*<Value>*$_{2.1}$	. . .	*<Value>*$_{2.n}$	*<Comment>*$_2$
		. . .		
<Constraint Name>$_m$	*<Value>*$_{m.1}$	. . .	*<Value>*$_{m.n}$	*<Comment>*$_m$

FIGURE D-35/X.290, Part 2

Compact Constraints Table Proforma

Example D-39 – Shows field names across the top of the table, and different instances of the PDU in rows within the table. However, where there are too may fields to fit conveniently across a page, it is permitted to transpose the table.

Example D-39 – Given a PDU X, with two fields P1 and P2, if the possible values for P1 and P2 are 0 and 1, then we have the table shown in Figure D-36/X.290, Part 2.

The Constraints Reference columns of in the Dynamic Part might then contain entries such as X(S1) and X(S4).

D.8.1.2 *Generic values*

Constraints may be parameterized using generic values. These shall be presented in the format shown in Figure D-37/X.290, Part 2. Again, an alternative compact version of the format may be used, this is given in Figure D-38/X.290, Part 2.

PDU Constraints			
PDU Name: X			
List Name	Field Name		Comments
	P1	P2	
S1	0	0	
S2	0	1	
S3	1	0	
S4	1	1	

FIGURE D-36/X.290, Part 2

Two field PDU example

Generic Value		
PDU Name: <*PDU Name*>	Generic Name: <*Generic Name*>	
Field Name	Value	Comments
.	.	.
.	.	.
.	.	.
<*Field Name*>	<*Value*>	<*Comment*>
.	.	.
.	.	.
.	.	.

FIGURE D-37/X.290, Part 2

PDU Constraint Proforma

Generic Value				
PDU Name: *<PDU Name>*	Generic Name: *<Generic Name>*			
List Name	Field Name			Comments
	<Field Name>$_1$	. . .	*<Field Name>*$_n$	
<Constraint Name>$_1$	*<Value>*$_{1.1}$	. . .	*<Value>*$_{1.n}$	*<Comment>*$_1$
<Constraint Name>$_2$	*<Value>*$_{2.1}$	. . .	*<Value>*$_{2.n}$	*<Comment>*$_2$
		. . .		
<Constraint Name>$_m$	*<Value>*$_{m.1}$	. . .	*<Value>*$_{m.n}$	*<Comment>*$_m$

FIGURE D-38/X.290, Part 2

Compact Generic Constraint Proforma

These concepts are illustrated in the Examples D-39 and D-40.

Example D-40 – The invocation of the PDU X (Figure D-39/X.290, Part 2) in a test step may be made as follows: X(S1), X(S2), X(S3), X(S4), X(S5(0)), X(S5(1)) or X(S5(Var)), where Var is either a variable (see § D.5.6 and D.8.2) or a field name of a received PDU (§ D.5.9) or an expression.

PDU Constraints			
PDU Name: X			
List Name	Field Name		Comments
	P1	P2	
S1	0	0	
S2	0	1	
S3	1	0	
S4	1	1	
S5(A)	1	A	

FIGURE D-39/X.290, Part 2

Simple Parameterized Constraint example

Example D-41 – The way of defining generic values illustrated in Example D-40 can be extended to a group of fields:

Given the PDU Y with three fields Q1, Q2 and Q3 with 0 and 1 as possible values, the table can be represented as shown in Figures D-40/X.290, Part 2, D-41/X.290, Part 2, D-42/X.290, Part 2.

PDU Constraint				
PDU Name: Y				
List Name	Field Name			Comments
	Q1	Q2	Q3	
T1	0	0	0	
T2(B1)	1	B1		
T3(B2)	B2		0	

FIGURE D-40/X.290, Part 2

Parameterized Constraint example (Part 1 of 3)

Generic Value			
PDU Name: Y	Generic Name: B1		
Constraint Name	Field Name		Comments
	Q3	Q4	
C1	0	0	
C2	0	1	

FIGURE D-41/X.290, Part 2

Parameterized Constraint example (Part 2 of 3)

Generic Value			
PDU Name: Y	Generic Name: B2		
Constraint Name	Field Name		Comments
	Q1	Q2	
D1	0	0	
D2	1	1	
D3	1	0	

FIGURE D-42/X.290, Part 2

Parameterized Constraint example (Part 3 of 3)

The invocation of the PDU Y in a test step is then made as follows: Y(T1), Y(T2(C1)), (T2(C2)), (Y(T2(D1))), etc.

Tables defining generic values shall not be referred to directly in the main behaviour part, but only via a parameterized constraint name (e.g., B1 is used as Y(T2(B1)) in Example D-41).

D.8.2 *Free variables*

D.8.2.1 *Introduction*

In many cases it is necessary to use the values received in a PDU (e.g., SRC-REF in a CC) for filling a corresponding field when sending another PDU (e.g., DST-REF in DT). Although this can be achieved by keeping track of the value in a variable from the dynamic part, it may also be convenient to represent this in a more concise way by using a Free Variable in the constraints part. Such a variable shall be declared at the beginning of the constraints part when declaring constraints. Examples D-40 and D-41 are not examples of free variables. In this case, one can represent that value by a free variable which is considered to be global to the Constraints Part.

These free variables shall be declared using the proforma shown in Figure D-43/X.290, Part 2.

A variable is bound to a value when a PDU in which it appears occurs as a parameter to a received ASP.

D.8.2.3 *Variables in constraints*

Whenever a constraint applies, any variable appearing in the field of a constraint either has, or takes on, the actual value sent or received.

a) If the variable is already bound to a value, either
   1) that value is sent; or
   2) that value must be the value received in that field if the constraint is to apply. (Otherwise the constraint does not apply.)

b) If the variable is unbound, either
   1) any value of the appropriate type may be sent; or
   2) the variable becomes bound to the received value as a result of applying the constraint.

A typical application of this feature might be to preserve addresses, or connection references from PDU to PDU. These may not be of interest in most test cases, but it is important that the same value is present in several different PDUs.

Free Variables		
Name	Type	Comments
.	.	.
.	.	.
.	.	.
<Name>	*<Type>*	*<Comments>*
.	.	.
.	.	.
.	.	.

FIGURE D-43/X.290, Part 2

Free Variables Proforma

D.8.2.4 *Conventions*

The conventions shown in Table D-1/X.290, Part 2 apply to constraint table entries.

TABLE D-1/X.290, Part 2

Constraint Table Entry Conventions

Direction	Field or Parameter Value	
	–	?
Initiated by Tester (!)	There shall be no explicit encoding of this field or parameter.	The value of this parameter has no influence on the outcome of the test case and so may be anything that is legal according to the relevant service or protocol standard*.
Received by Tester (?)	There shall be no explicit encoding of this field or parameter.	The tester need make no additional verification of this field or parameter.

Note – A test case writer should be aware that if an ASN-1 parameter has a default value, then according to Recommendation X.209, an IUT is entitled to encode the default value explicitly, or to omit the parameter from the encoding.

D.8.2.4.1 *Pattern matching in a Character String*

Inside a character string, a ? in place of a character means that any single character is accepted. When the ? symbol itself is needed within the character string, this shall be indicated by preceding the ? with the special character \. The character \ itself is written \\.

Inside a character string a * means that none, or any number, of characters is acceptable. This * shall match the longest sequence of characters possible, according to the pattern as specified by the symbols surrounding the *. (When the * symbol itself is needed within the character string, this shall be indicated by preceding the * with the special character \.)

D.8.3 *ASN-1 modular method*

D.8.3.1 *Scope of method*

This method assumes that ASN-1 descriptions of the PDUs or ASP parameters are available, and takes them as a convenient basis for constraint description.

D.8.3.2 *Description of the ASN-1 modular method*

When a PDU or an ASP parameter is defined in the declaration part using ASN-1, the ASN-1 modular method is applicable. This method provides the following major facilities:

a) **Specification**: the (partial) specification of ASN-1 values. Typically, there will be correct PDUs, but arbitrary ASN-1 values may be constructed and used.

   Note – ASN-1 requires explicit values, whereas the modular method allows "don't care".

b) **Naming**: these values may be named, so that they can be referenced from the dynamic part.

c) **Annotation**: values may be annotated with an extended form of ASN-1 type notation. Value encodings may be specified so that two identical values encoded differently may be distinguished. Constraints may be placed on components of values through the use of free variables, or the – and ? conventions, just as constraints can be placed on field values in the tabular method.

d) **Replacement**: new values may be constructed from old ones by replacing components of existing values with new values.

D.8.3.3 *Value specification*

When specifying an ASN-1 value the following auxiliary information shall be provided:

a) A name.

   This is required to identify the value.

   When specifying large, complex ASN-1 values, it is sometimes convenient to be able to place constraints on selected components of a value only. Such components can be defined by tracing a path through the ASN-1 type declaration corresponding to the value, identifying elements at each level by name and using a period to separate each element name.

b) The ASN-1 type of the value.

   This is required for documentation purposes so that the ASN-1 value can be related to an appropriate ASN-1 type definition to be found in the relevant protocol standard.

   In order to make an ASN-1 type definition suitable for documenting ASN-1 value definitions, a number of abbreviations and restrictions to the ASN-1 type notation are required. These are enumerated below. Some of these represent changes to the grammar to be found in (Recommendation X.208, Annex F).

   The following notation is used:

```
<NonTerminal> ::=
          . . .
        | <NewProds>
```

   where *<NonTerminal>* is defined in (Recommendation X.208, Annex F) and ... represents that definition. *<NewProds>* represents the TTCN extensions.

   1) It is permitted to omit the *<Type>* from an *<identifier>* *<Type>* combination. The following productions are added to (Recommendation X.208, Annex F):

```
<ElementType> ::=
          . . .
        | <TTCNElementType>
<TTCNElementType> ::=
        <TTCNNamedType>
<TTCNNamedType> ::=
        <identifier>
        | <NamedType>
```

2) In a *<ChoiceType>*, only the type of the choice made shall be used. However, in order to remind the reader that a choice has been made, the CHOICE keyword may be retained. The following production replaces the production for *<ChoiceType>* in (Recommendation X.208, Annex F):

```
<ChoiceType> ::=
        <TTCNNamedType>
        | CHOICE <TTCNNamedType>
```

3) The *<SelectionType>* shall not be used. Instead, the actual type selected shall be used. *<SelectionType>* is removed from the list of possible *<BuiltinTypes>* in (Recommendation X.208, Annex F).

4) The *<SequenceOfType>* is changed to permit the naming of the actual values to be sent out. The following production replaces the production for *<SequenceOfType>* in (Recommendation X.208, Annex F):

```
<SequenceOfType> ::= SEQUENCE OF
        { <SequenceOfTypeList>}
<SequenceOfTypeList> ::=
        ...
        | <SequenceOfTypeList>
        <SequenceOfTypeType>
<SequenceOfTypeType> ::=
        <Type>.<number>
```

All the Types appearing in a *<SequenceOfTypeList>* shall be identical. The numbers in a *<SequenceOfTypeList>* shall be distinct. They are used to index the Types.

c) A specification of encoding constraints on the value.

If any legal encoding of the value is acceptable, the specification of encoding constraints for the value may be omitted. Otherwise, the encoding specification shall consist of two parts:

1) A specification of the *<Identifier>* appearing in the encoding of the value. The *<Identifier>* shall consist of a whole number of octets.

```
<Identifier> ::= [ <IdentifierValue> ]
<IdentifierValue> ::=
        <hstring>
        | <bstring>
        | <variable>
```

It is not necessary for the *<IdentifierValue>* to be correct with respect to the ASN-1 type specification. This means that incorrectly encoded PDUs may be specified.

2) A specification of the *<Length>*, to be used in the encoding of the value. The *<Length>* shall consist of a whole number of octets.

```
<Length> ::= [ <LengthValue> ]
<LengthValue> ::=
        | <hstring>
        | <bstring>
        | <variable>
        | LI
        | SD
        | LD
        | LD <number>
        | IN
```

LI specifies that any legal encoding of the correct length may be used.

SD specifies that the *Short Definite* length type shall appear in the encoding.

LD specifies that the *Long Definite* length type shall appear in the encoding. The length shall be padded out to *<number>* octets if *<number>* appears.

IN specifies that *Indefinite* length type shall appear in the encoding.

3) A specification of the value itself.

```
<Value> ::= [ <TTCNNamedValue> ]
<TTCNNamedValue> ::=
        | <NamedValue>
        | <valuereference>
        | ?
        | –
        | <in> <identifier>1
          REPLACE <identifier>2
          BY <TTCNNamedValue>
<in> ::= IN | <empty>
<BuiltinValue>::=
        ...
        | <TTCNNamedValue>
```

Values are specified using the standard ASN-1 value notation with the following extensions:

A) Values defined in either the Test Suite Parameters § D.5.4, or elsewhere in the Constraints Part may be referenced.

B) < *CharacterString*> values may be denoted by enclosing the string in double quotes. Within those quotes a pair of consecutive double quotes shall be used to denote the double quote itself.

C) A ? is used to specify "don't care" values. A ? appearing in a value that is to be sent by the tester may take any value that is legal in that context (according to the relevant service or protocol standard). A tester need make no verification of received values specified as ?.

D) A – is used to specify that the encoding of the value shall be absent.

E) The replacement operation provides the ability to substitute ASN-1 values (and their corresponding type and encoding constraints) for components of existing values, thereby producing new values, with the meaning *replace the component named <identifier>*$_1$ *of the value <identifier>*$_1$ *with the <TTCNNamedValue>*.

This information shall be provided in the format shown in Figure D-44/X.290, Part 2. The drawing of the box lines is optional in this case.

ASN-1 Value Constraint Declaration				
ASN-1 Type	Identifier	Value	Value	Comments
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
<ASN-1 Value Constraint Name> *<ASN-1 type>*	*<Identifier>*	*<Length>*	*<Value>*	*<Comments>*
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.

FIGURE D-44/X.290, Part 2

ASN-1 Value Constraint Declaration Proforma

Example D-42 – This example shows how to reference PDU values from the dynamic part. We define a PDU called APDU:

```
APDU ::= SEQUENCE {
      INTEGER
      BOOLEAN
      }
```

and the global variable N. An instance of APDU might be:

```
Expected PDU ::= SEQUENCE {
      INTEGER [10] [LI] N
      BOOLEAN [ID] [LI] TRUE
      }
```

Then in the behaviour description we can write:

?DATAind [(UserData ExpectedPDU) AND (N = 27)]

APPENDIX I

(to Recommendation X.290, Part 1)

Applicability of the test methods to OSI* protocols

I.1 *Introduction*

Physical layer and media access control protocols are outside the scope of this Recommendation. None of the four methods defined in this Recommendation is intended to be applicable to these protocols.

I.2 *Data link protocols*

For testing data link protocols, the following points should be considered:

a) The local single-layer test method is applicable only if the physical layer boundary of the IUT is accessible. In practice this is unlikely to be the case, except when testing a prototype or the design of an algorithm before it is put into hardware or firmware.

b) Since the physical service is not end-to-end, the distributed, coordinated and remote single-layer test methods are applicable only when the external lower tester is connected to the SUT by a single link rather than across a network.

c) The distributed, coordinated and remote single-layer test methods are applicable only if a lower tester can be realized with control over physical service primitives (or perhaps more realistically physical and data link PDUs). This may be difficult for some types of subnetwork.

d) Most protocols for real subnetworks were defined before physical or data link services were thought of. Hence, service boundaries below the network service are generally either unavailable or not well-defined. Hence, it is generally felt that the control and observation for data link layer testing needs to be interpreted in terms of PDUs rather than ASPs. However, there may be some systems which do provide a data link service boundary or similar technology-dependent interface.

If single-layer testing of a data link protocol is not possible, multi-layer or single-layer embedded methods should be considered.

I.3 *Network protocols*

For network protocols, the test methods to be used are dependent upon whether the IUT is an end-system or an open relay system.

Since services below the network service are generally either unavailable or not well-defined, it is generally felt that the control and observation for network layer testing needs to be interpreted in terms of PDUs rather than ASPs.

It should be recognized that with some subnetwork technologies there are more than three protocols required to provide the network service. Each of these protocols may be tested separately or in any combination of adjacent protocols.

Considering the layer as a whole, both network and data link abstract service primitives are controllable and observable. Thus, for end-systems, all four single-layer (non-embedded) test methods are applicable, but since the data link service is not end-to-end, the lower tester has to be connected to the SUT over a single link for the distributed, coordinated and remote single-layer test methods.

Both the loop-back and transvers test methods are applicable to testing network relay systems.

I.4 *Transport protocol*

All the abstract test methods defined in this Recommendation are applicable to transport protocol conformance testing.

I.5 *Session protocol*

All the abstract test methods defined in this Recommendation are applicable to session protocol conformance testing.

For a large group of systems it will be appropriate to test the session protocol in combination with presentation and application protocols. Testing of session protocol should, therefore, normally be done in one of the two following ways:

a) as a single-layer implementation or in combination with underlying protocols, in order to test the provision of a general purpose session service capable of supporting several different applications; the distributed or coordinated single-layer test methods are likely to be appropriate;

b) in combination with presentation and application protocols, in order to test it in a specific application context; the remote or distributed single-layer embedded test methods are likely to be appropriate.

I.6 *Presentation and application protocols*

I.6.1 *General comments*

Conformance tests can be specified abstractly in terms of service primitives, irrespective of whether there is any notion of a service access point associated with them. Thus, provided that there is some mapping between application service primitives and real effects which can be observed and/or controlled, then tests can be specified in terms of those application service primitives. The observation and control of the service primitives may be indirect because of the nature of the mapping onto real effects, but as long as that mapping is possible then tests specified in these terms can be run. It is hard to find an example of an application service primitive which can never have a realization that can be observed and controlled; indeed if there were any, it seems likely that they ought not to be defined as primitives at all.

It is accepted that, in some circumstances, application protocol standards or recommendations may specify requirements on real effects which have to be achieved as the result of protocol exchanges (in particular this is evident in Job Transfer and Manipulation (JTM)). However, these requirements on real effects should be kept quite distinct from the normal protocol conformance requirements, possibly even in separate (perhaps functional) standards or recommendations. Some of these "non-protocol" conformance requirements might be testable by means of the general purpose abstract test methods defined in this Recommendation, but in general they will require application-specific test methods which fall outside the scope of this Recommendation.

I.6.2 *Association control*

Association control is unusual in that it involves 3 phases, the second of which is defined by another ASE. Association control service primitives could be observed and controlled in some systems. If so, it would be possible to test the association control protocol (possibly in combination with the presentation protocol) in isolation from any other ASE. The isolated testing of association control would have to use a dummy ASE for testing purposes. Any of the test methods could be used, including the coordinated method with a test management protocol defined as a dummy ASE. However, such isolated testing is of limited value as it could only test the protocol machine, leaving untested the mapping between abstract and transfer syntaxes. This aspect can only be tested for a particular ASE. The use of a dummy ASE in testing association control would not in general involve the syntax mappings that other real ASEs would. Therefore, priority should be given to testing association control in combination with the presentation protocol and another real ASE, using the remote or distributed single-layer embedded test methods. In other words, the emphasis should be on testing application context operations.

The association control protocol has no non-protocol conformance requirements.

I.6.3 *Virtual terminal (VT)*

Virtual terminal protocol can be used in three ways: terminal to VT host, terminal to terminal, and VT host to VT host. In all three cases the VT service primitives could be observable and controllable in a large enough proportion of systems to make their use in test specification realistic. VT hosts may provide VT service interfaces in order to permit implementation of arbitrary user processes, so such an interface can be used by a realization of an upper tester. Terminals, on the other hand, are most likely to provide control and observation of VT service primitives only indirectly through a user interface which may be very different from the VT service. Some terminals might, in addition, provide direct access to a VT service interface.

Hence, the remote and distributed single-layer test methods are applicable to VT protocol testing.

I.6.4 *File transfer access and management (FTAM)*

In practice, observation and control of service primitives is different for FTAM initiators and responders. In general, they can be observed and controlled when testing initiators, but not when testing responders. The problem with FTAM responders is that there will be real effects associated with their service primitives, but the observation and control of these will be highly system specific. This is because although the "user" of the FTAM service in a responder is the virtual filestore, within the system under test effects on the VFS can only be seen through the real filestore. Thus, the remote single-layer test method is applicable to both FTAM initiators and responders, but the distributed single-layer test method is applicable only to FTAM initiators.

The testing of file management can be carried out on its own if necessary and does not depend on which of the read and write functional units are implemented. The testing of file transfer and access does, however, depend on which of these functional units are implemented.

The testing of file transfer and access for read-only systems or read/write systems is relatively straightforward. Read/write systems can be tested by transferring files to the system and then reading them back again. Although there is in general nothing to prevent local modification of such files between reading and writing, such local modification could be prevented for the duration of the test suite execution, by keeping other users off the system. Read-only systems can be tested by pre-loading the required set of files by local means and then running tests to read them using FTAM. In the testing of write-only systems, the assignment of verdicts to outcomes has to rely on a highly system specific (possibly subjective) interpretation of what the outcome (i.e. detailed record of observed test events) is.

Another way of viewing this situation is to recognize that FTAM has non-protocol conformance requirements concerned with the real effects which should occur as the result of PDU exchanges. These requirements can be tested by general test methods provided that the methods involve observation of FTAM service primitives (i.e. the test execution involves observation of the real effects associated with such primitives). In order for the real effects to be observed during testing, the test laboratory will need to be supplied with a detailed description of the mapping of service primitives onto real effects. This amounts to requiring the whole real filestore definition. This information should therefore be referenced by the PIXIT, being more information than one would want to include explicitly in the PIXIT or in the PICS.

There is a conformance requirement that, for the purposes of testing, an FTAM implementation shall be usable without the private use or legal qualification attributes. Nevertheless, there are other attributes which are difficult or practically impossible to test. For example, data protection and security attributes come in this category, because the information needed to test them will probably not be available. Also the ability to use the storage attributes to lock a file against non-OSI access will probably be impossible to verify, because the fact that the test laboratory cannot access such a file says nothing about the capabilities of a user with greater access permissions.

I.6.5 *Job transfer and manupulation (JTM)*

The situation regarding observation and control of service primitives is similar to that for FTAM. For JTM initiators, observation and control of service primitives is likely to be possible, and thus both the remote and distributed single-layer test methods are applicable. For JTM responders, the distributed single-layer test method can be used only if the real effects associated with the service primitives can be observed. This will involve knowing a mapping which may be complex. However, the JTM protocol standard places requirements on the implementor to state what these effects are and to make certain information human readable. Typically these effects are concerned with various documents being created and moved around. Thus, it may be possible to use the distributed single-layer test method for JTM initiators as well as the remote single-layer test method.

Status/enquiry primitives are easily testable. Reports on progress of a job can be observed.

A multi-system test method is needed to test major aspects of JTM. All but one system could be (lower) testers, and several logically distinct (lower) testers could be realized in the same real system. Nevertheless, there needs to be more than one external (lower) tester from an abstract point of view. For example, tests of failure situations need to cover the situation in which one system crashes and the remaining systems are expected to recover. It may also be desirable to test a collection of systems rathar than a single system. Thus, at a minimum, there could be two new sets of test methods, one involving a single SUT and 2 (lower) testers, the other involving 2 SUTs and a single (lower) tester. Such multi-system test methods are outside the scope of this Recommendation.

Note – A similar requirement for a multi-system test method exists for MHS.

I.6.6 *Message handling protocols*

There are CCITT test suites for three protocols, the inter-personal message service (IPMS) (P2 protocol), the message transfer service (MTS) (P1 protocol), and reliable transfer service (RTS). There are also two basic system configurations that require consideration.

The first is the "end-system test configuration" which can be used to test P2, P1 and RTS. The second is the "relaying message transfer agent test configuration" which can be used to test the relaying aspects of the P1 protocol.

The P2 protocol test suite uses one external PCO at the boundary between the user agent layer and the MTS and one PCO at the upper boundary of the IUT. However, the P2 protocol does not include definitions of ASPs, so it has therefore been necessary to construct "hypothetical" ASPs in order to describe the abstract test suite in a formal way.

The P1 protocol test suite for end-system testing uses one external PCO at the boundary between the message transfer layer and the RTS and one PCO at the upper boundary of the IUT. However, testing of multiple destination delivery requires more than one user agent, which in turn requires more than one PCO at the upper boundary of the IUT. The P1 protocol test suite for relay testing uses two external PCOs at the boundary between the message transfer layer and the RTS.

The RTS test suite uses one external PCO at the boundary between the reliable transfer layer and the session layer and one PCO at the upper boundary of the user agent layer within the SUT. The RTS test suite also includes a third service access point in the description of the test cases; this is between the message transfer and the RTS within the SUT; it is used for clarification only and is not a PCO.

Therefore, the distributed single-layer test method is used for end-system testing of the P1 and P2 protocols; the distributed single-layer embedded test method is used for testing RTS; and the transverse test method is used for testing the relaying aspects of the P1 protocol. In all cases of end-system testing, the remote single-layer test method is also applicable.

A further point is that the abstract test suites for P2 and P1 protocols include X.409 encoding and decoding tests.

Note – The test suites referred to here are concerned with testing implementations of the 1984 versions of the CCITT X.400-Series Recommendations.

I.6.7 *Commitment concurrency and recovery (CCR)*

For reasons similar to those given in I.6.2 for association control, CCR should be tested as part of an application context, such as that required by JTM. Thus, the remote and distributed single-layer embedded test methods are applicable.

CCR contains a requirement to preserve secure data beyond crashes, although it recognizes that this is not possible to implement in practice because some error situations (e.g. bomb on the installation) are too catastrophic. In order to test this requirement to preserve secure data beyond crashes, it is necessary for the implementor to state in either the PICs or PIXIT which error situations would not affect the preservation of secure data.

I.6.8 *Presentation*

The service primitives are potentially observable and controllable to the same extent as for lower layers. Thus, all four single-layer (non-embedded) test methods are theoretically applicable. However, as discussed above in relation to association control, the testing of presentation protocol in isolation from an ASE is of limited value, because it could only test the protocol machine, leaving untested the more interesting aspect of the presentation layer, namely the mapping between abstract and transfer syntaxes. Therefore, the testing of presentation protocol embedded under association control and another ASE is preferred. Thus, the relevant applicable test methods are the remote and distributed single-layer embedded ones.

I.6.9 *Transfer syntaxes*

Transfer syntaxes (e.g. ASN-1 or X.409) are rather different from the OSI* protocol Recommmendations* with respect to conformance. In general, there would not be conformance testing of the encoding rules of a transfer syntax independent of the application protocol using those rules.

It is noted that there are, for instance, conformance requirements in the ASN-1 basic encoding rules Recommendation. The requirement that where alternative encodings are provided as a sender's option, conforming receivers shall support all alternatives does require ASN-1 specific conformance testing. Nevertheless, the ASN-1 basic encoding rules will always be tested with the presentation protocol and the test methods appropriate to that protocol will be chosen.

I.7 *Management protocols*

Since system management and application management reside in the application layer, the general comments above which apply to application protocols, also apply to system and application management protocols. In particular, test cases for such management protocols can be specified in terms of management service primitives, provided that there is some mapping between the management service primitives and real effects which can be observed and/or controlled. If such service primitives and such a mapping exist, then the testing can be performed by the distributed test method. Otherwise, the testing can be performed by using the remote test method.

Also, as with other application protocols, the exchange of PDUs of a system management protocol can be tested by using the test methods described in this Recommendation. The application of conformance testing to other aspects of application and system management services and related "non-protocol" activities of management entities is, however, outside the scope of this Recommendation.

Thus, because of the "non-protocol" conformance requirements associated with system management protocols, in particular, the testing of conformance to them cannot fully be achieved by means of the methods described in this Recommendation. For example, the correct operation of the system management function of modifying the information in a directory cannot be tested merely by exchanging PDUs.

Furthermore, system management functions are not only related to the application layer but also to the operations of the underlying protocols. Therefore, the testing of the system management protocol needs to be carried out using an implementation of the underlying protocols which has already been tested.

The testing of layer management protocols is dependent upon the existence of the relevant protocols but, given that these exist, it should be possible to test them by means of the test methods described in this Recommendation.

I.8 *Connectionless protocols*

Since each test method described in this Recommendation is defined in terms of observation and control of ASPs and PDUs, and not in terms of connections, then all methods are applicable to the testing of connectionless protocols, taking into account the restrictions applying to each layer.

APPENDIX II

(to Recommendation X.290, Part 1)

Index of definitions of terms

APPENDIX III

(of Recommendation X.290, Part 2)

Examples for guidance for PICS proforma specifiers

III.1 *Abbreviations*

The following examples use the abbreviations: "m" for mandatory "c" for conditional, "o" for optional, "n" for negotiable and "–" for not applicable, as defined in § 7.1.7.

III.2 *PDU example*

PDU NAME	CLAUSE REFERENCE	SUPPORT	
		TRANSMIT	RECEIVE
PDU-1	a.b.c	m	m
PDU-2	a.b.d	o	m
PDU-3	a.b.e	c	–

III.3 *Parameter example*

PARAMETER	CLAUSE REFERENCE	PERMITTED RANGE OF VALUES	VALUES SUPPORTED	SUPPORT STATUS
PARM-1	x.y & p.q	unlimited		n
PARM-2	f.g.h	1 → 10		o
PARM-3	f.i	4		c
PARM-4	w.x.y.z	type-A		m

A variant of this example would be a timer example which might include an extra column for units.

III.4 *Supported services example*

OPTION	Clause	SUPPORTED ON			
		2 Cans and Wet String	Carrier Pigeon	ExtraSensory Perception	UltraSound
Whisper	4	m	–	o	m
Echo	5	n	o	c	n
Whistle	6	o	–	c	m

APPENDIX IV

(to Recommendation X.290, Part 2)

Example of choice of abstract test methods

IV.1 *Rationale for the transport layer*

Objective: test a transport entity (single-layer IUT) in a real open system.

Assumption 1: the real open system is used for several applications.

Assumption 1.1: all applications use the OSI transport service via a locally defined and accessible interface.

Assumption 1.2: all applications use the OSI transport service via a locally defined and accessible session service interface, the transport service boundary being inaccessible.

Assumption ·1.3: all applications use the OSI transport service via the OSI session service, but neither the transport service boundary nor the session service boundary are accessible.

Assumption 2: the real open system is used for one application only.

Assumption 2.1: all layer boundaries are accessible via locally defined interfaces.

Assumption 2.2: no layer boundary is accessible – the system does not provide interfaces except for the end-user (this is the case for Teletex and MHS monolithic products).

Assumption 2.3: no layer boundary is accessible – the system does not provide interfaces – even the end-user cannot access the boundary between the OSI and non-OSI of the application process.

The external test methods apply to these assumptions as follows:

1.1	directly	CS	or	DS		
1.2	embedded via session layer	CSE	or	DSE		
1.3		RS				
2.1	directly	CS	or	DS	or	LS
2.2	embedded via all upper layers	CSE	or	DSE		
2.3		RS				

Conclusions: CS, CSE, DS, DSE, RS and LS test methods all apply to the transport layer. Embedded test methods can be used via the session layer on its own or via all the upper layers.

Priority: assumptions 1.1, 1.2 and 2.2 appear to be the most common cases (respectively, the direct access method, session access method, and Teletex and MHS monolithic products). Therefore, test methods CS, CSE and DSE should be given the highest priority.

IV.2 *Main test methods to be used for the transport layer*

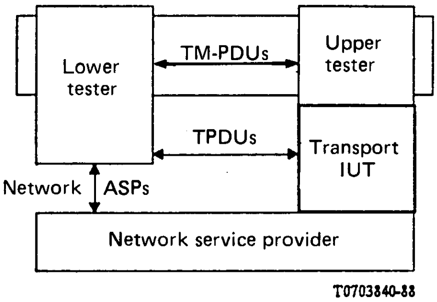


FIGURE IV-1/X.290, Part 2

The CS test method

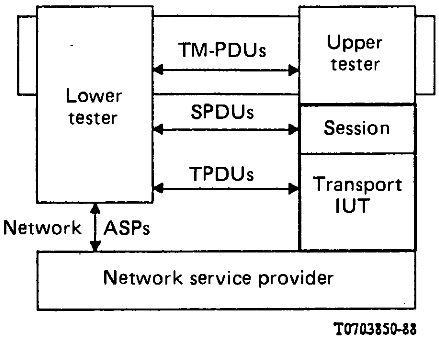


FIGURE IV-2/X.290, Part 2

The CSE test method

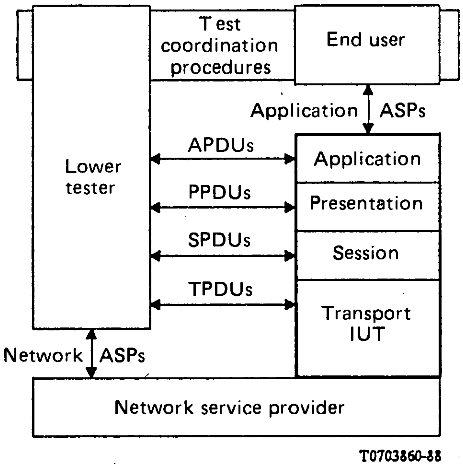


FIGURE IV-3/X.290, Part 2

The DSE test method

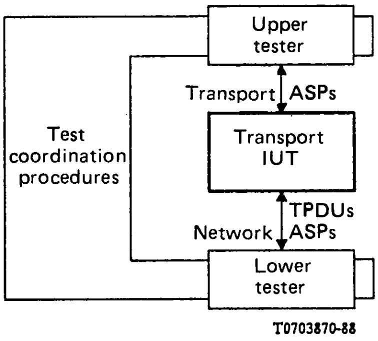


FIGURE IV-4/X.290, Part 2

The LS test method

ISBN 92-61-03701-1