INTERNATIONAL TELECOMMUNICATION UNION

CCITT

THE INTERNATIONAL
TELEGRAPH AND TELEPHONE
CONSULTATIVE COMMITTEE

BLUE BOOK

VOLUME VIII – FASCICLE VIII.4

DATA COMMUNICATION NETWORKS

OPEN SYSTEMS INTERCONNECTION (OSI) MODEL AND NOTATION, SERVICE DEFINITION

RECOMMENDATIONS X.200-X.219

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.4

DATA COMMUNICATION NETWORKS

OPEN SYSTEMS INTERCONNECTION (OSI) MODEL AND NOTATION, SERVICE DEFINITION

RECOMMENDATIONS X.200-X.219

IXTH PLENARY ASSEMBLY
MELBOURNE, 14-25 NOVEMBER 1988

Geneva 1989

ISBN 92-61-03691-0

Printed in Spain

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

BLUE BOOK

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

FASCICLE X.1 – Functional Specification and Description Language (SDL). Criteria for using Formal Description Techniques (FDTs). Recommendation Z.100 and Annexes A, B, C and E, Recommendation Z.110 (Study Group X).

FASCICLE X.2 – Annex D to Recommendation Z.100: SDL user guidelines (Study Group X).

FASCICLE X.3 – Annex F.1 to Recommendation Z.100: SDL formal definition. Introduction (Study Group X).

FASCICLE X.4 – Annex F.2 to Recommendation Z.100: SDL formal definition. Static semantics (Study Group X).

FASCICLE X.5 – Annex F.3 to Recommendation Z.100: SDL formal definition. Dynamic semantics (Study Group X).

FASCICLE X.6 – CCITT High Level Language (CHILL). Recommendation Z.200 (Study Group X).

FASCICLE X.7 – Man-Machine Language (MML). Recommendations Z.301-Z.341 (Study Group X).

CONTENTS OF FASCICLE VIII.4 OF THE BLUE BOOK

Recommendations X.200 to X.219

Open Systems Interconnection (OSI);
Model and Notation, Service Definition

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 the 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 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/IEG standards are given in the table below.

CCITT Recommendation	ISO/IEC Standard or technical report
X.200	ISO 7498, Information processing systems – Open Systems Interconnection – Basic Reference Model (1984).
X.208	ISO 8824, Information processing systems – Open Systems Interconnection – Specification of Abstract Syntax Notation One (ASN.1) (1987).
	ISO 8824/AD1, Information processing systems – Open Systems Interconnection – Specification of Abstract Syntax Notation One (ASN.1) – Addendum 1: ASN.1 Extensions [a].
X.209	ISO 8825, Information processing systems – Open Systems Interconnection – Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1) (1987).
	ISO 8825/AD1, Information processing systems – Open Systems Interconnection – Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1) Addendum 1 : ASN.1 Extensions [a].
X.210	ISO TR 8509, Information processing systems – Open Systems Interconnection – Service conventions (1987).
X.211	ISO 10022, Information processing systems – Open Systems Interconnection – Physical service definition [b].
X.212	ISO 8886, Information processing systems – Data communication – Data link service definition for Open Systems Interconnection [b].
X.213	ISO 8348, Information processing systems – Data communications – Network service definition (1987).
	ISO 8348/AD2, Information processing systems – Data communications – Network service definition – Addendum 2: Network layer addressing (1988).
	ISO 8348/AD3, Information processing systems – Data communications – Network service definition – Addendum 3: Additional features of the network service (1988).
X.214	ISO 8072, Information processing systems – Open Systems Interconnection – Transport service definition (1986).
X.215	ISO 8326, Information processing systems – Open Systems Interconnection – Basic connection oriented session service definition (1987).
	ISO 8326/AD2, Information processing systems – Open Systems Interconnection – Basic connection oriented session service definition – Addendum 2: Incorporation of unlimited user data service [a].
X.216	ISO 8822, Information processing systems – Open Systems Interconnection – Connection oriented presentation service definition (1988).
X.217	ISO 8649, Information processing systems – Open Systems Interconnection – Service definition for the Association Control Service Elements [c].
X.218	ISO 9066-1, Information processing systems – Text communication – Reliable Transfer – Part 1: Model and service definition [b].
X.219	ISO 9072-1, Information processing systems – Text communication – Remote Operations – Part 1: Model notation and service definition [b].

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

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

[c] Presently awaiting publication.

FASCICLE VIII.4

Recommendations X.200 to X.219

DATA COMMUNICATION NETWORKS: OPEN SYSTEMS INTERCONNECTION (OSI); MODEL AND NOTATION, SERVICE DEFINITION

PAGE INTENTIONALLY LEFT BLANK

PAGE LAISSEE EN BLANC INTENTIONNELLEMENT

SECTION 1

MODEL AND NOTATION

Recommendation X.200

REFERENCE MODEL OF OPEN SYSTEMS INTERCONNECTION FOR CCITT APPLICATIONS[1)]

The CCITT,

considering

(a) that Administrations in many countries are planning to establish telecommunication services which exploit the networks now existing or due to be available in the near future;

(b) that those services may be carried on different types of networks;

(c) that users of these services need to communicate with each other irrespective of the diversity of interconnected networks;

(d) that methodical representation of network services will further promote the effective and efficient use of networks;

(e) that accommodation of conflicting service requirements and network constraints can be effected through the analysis of service and network functions. This analysis should lead to a universally applicable logical structure which may be used in the development of compatible service, interface and procedure definitions;

(f) that this structure should as far as possible accommodate existing Recommendations to allow the steady evolution of networks providing the new services;

(g) that close collaboration and liaison with other bodies studying reference models is desirable to ensure the widest possible applicability of resulting Recommendations,

unanimously recommends

that for CCITT applications:

(1) the methodical representation of new services be undertaken in accordance with the principles and architecture of this Recommendation;

(2) the structure of this reference model contained within this Recommendation be employed in the specification of new interface and procedural definitions;

(3) the user and Administration requirements for both services and management of services can be satisfied by application of the principles of this Recommendation.

[1)] Recommendation X.200 and ISO 7498 [Information Processing Systems – Open Systems Interconnection – Basic Reference Model] were developed in close collaboration and are technically aligned.

CONTENTS

0 Introduction

0.1 *About this Recommendation*

This Recommendation presents the purpose, framework and function of a reference model structure (hereinafter called "the Reference Model") for the logical process in a communications system. Instances of communication system elements that have been defined using this Reference Model may be found in other Recommendations.

Communications systems which employ the standardized communication procedures and methods derived from the Reference Model are referred to as "open systems", and such interconnection is referred to as "Open Systems Interconnection" (OSI). This Reference Model is consistent with the principles established by ISO for Open Systems Interconnection.

This Recommendation allows standardized procedures to be defined enabling the interconnection and subsequent effective exchange of information between users. Such users are systems; i.e., a set of one or more computers, associated software, peripherals, terminals, human operators, physical processes, information transfer means, etc., that form an autonomous whole capable of performing information processing and/or information transfer. The Reference Model, in particular, will permit interworking between different networks, of the same or different types, to be defined such that communication may be achieved as easily over a combination of networks as over a single network.

The term "user" implies nothing in respect to the contractual customer-Administration relationship; a user may be either outside the Administration or part of the Administration.

Conformance with the Reference Model does not imply any particular implementation or technology, but rather refers to the support of standardized information exchange procedures derived according to its provisions as specified in this Recommendation. The Recommendation provides neither details nor definitions or interconnection protocols.

The Reference Model serves as a framework for the definition of services and protocols which fit within the boundaries established by the Reference Model. In those few cases where a feature is explicitly marked as "optional" in the Reference Model, it should remain optional in the corresponding service or protocol (even if at a given instant the two cases of the option are not yet documented).

1 Scope and field of application

1.1 *The scope of the Reference Model*

a) to specify a universally applicable logical structure encompassing the requirements of CCITT applications;

b) to act as a reference during the development of new communications services, including potential CCITT recommended services, and the definition of the corresponding procedures;

c) to enable different users to communicate with each other by encouraging the compatible implementation of communication features;

d) to enable the steady evolution of CCITT applications by allowing sufficient flexibility so that advancements in technology and expanding requirements of users can be accommodated;

e) to allow comparison of a proposed new user requirement with the services for existing user requirements, thus allowing the new requirements to be satisfied in a manner compatible with existing CCITT recommended services.

1.2 *The application of the Reference Model*

This model will be employed in the development of interconnection protocols for communicating services, including potential CCITT recommended services, as follows:

a) A new user requirement is first expressed in user oriented terms. The requirement is then analyzed to determine the underlying patterns which would allow it to be grouped into functional subsets;

b) While the specification of a requirement will contain much narrative text for clarification, there may also be a requirement formally stated using a formal description technique (FDT);

c) A set of service definitions and protocol specifications is being written for each layer. Extensions and new uses of OSI will be progressively incorporated as new CCITT X-series Recommendations;

d) New functions that are identified will be incorporated into the Reference Model to enhance its future applicability;

e) For new uses and applications of OSI where no appropriate protocol is contained in the Recommendations, new protocols, particularly for the application layer, will be needed.

2 Definitions

Definitions of terms are included at the beginning of individual divisions and sub-divisions. An index of these terms is provided in an Annex B for easy reference.

3 Notation

Layers are introduced in division 5. An (N)-, (N + 1)-, and (N − 1)-notation is used to identify and relate adjacent layers:

(N)-Layer: any specific layer;

(N + 1)-Layer: the next higher layer;

(N − 1)-Layer: the next lower layer.

This notation is also used for other concepts in the model which are related to these layers, e.g. (N)-protocol, (N + 1)-service.

Division 6 introduces names for individual layers. When referring to these layers by name, the (N)-, (N + 1)- and (N − 1)-prefixes are replaced by the names of the layers, e.g. transport-protocol, session-entity and network-service.

4 Introduction to Open Systems Interconnection (OSI)

Note – The general principles described in divisions 4 and 5 hold for all layers of the Reference Model, unless layer specific statements to the contrary are made in divisions 6 and 7.

4.1 *Definitions*

4.1.1 **real system**

A set of one or more computers, the associated software, peripherals, terminals, human operators, physical processes, information transfer means, etc., that forms an autonomous whole capable of performing information processing and/or information transfer.

4.1.2 **real open system**

A real system which complies with the requirements of OSI Recommendations in its communication with other real systems.

4.1.3 **open system**

The representation within the Reference Model of those aspects of a real open system that are pertinent to OSI.

4.1.4 **application-process**

An element within a real open system which performs the information processing for a particular application.

4.2 *Open Systems Interconnection environment*

In the concept of OSI, a real system is a set of one or more computers, associated software, peripherals, terminals, human operators, physical processes, information transfer means, etc., that forms an autonomous whole capable of performing information processing and/or information transfer.

An application-process is an element within an open system which performs the information processing for a particular application.

Application-processes can represent manual processes, computerized processes or physical processes. Some examples of application-processes that are applicable to this open system definition are the following:

a) a person operating a banking terminal is a manual application-process;

b) a FORTRAN program executing in a computer centre and accessing a remote database is a computerized application-process; the remote database management systems server is also an application-process; and

c) a process control program executing in a dedicated computer attached to some industrial equipment and linked into a plant control system is a physical application-process.

OSI is concerned with the exchange of information between open systems (and not the internal functioning of each individual real open system).

As shown in Figure 1/X.200, the physical media for open systems interconnection provides the means for the transfer of information between open systems.

Note – At this point, only telecommunications media have been considered. The use of other interconnection media is for further study.

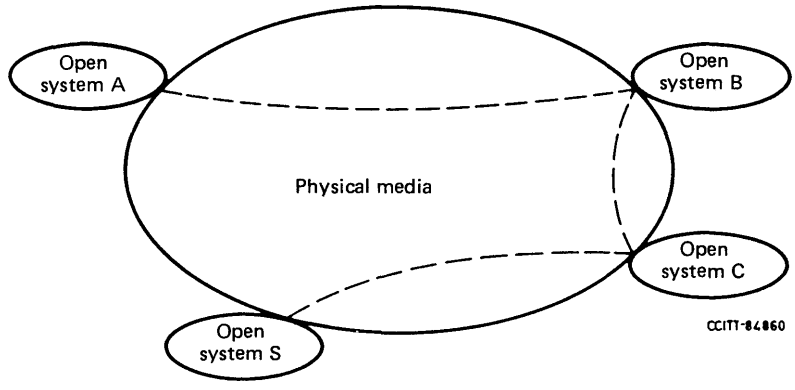


FIGURE 1/X.200

Open systems connected by physical media

OSI is concerned only with interconnection of systems. All other aspects of systems which are not related to interconnection are outside the scope of OSI.

OSI is concerned not only with the transfer of information between systems, i.e. transmission, but also with their capability to interwork to achieve a common (distributed) task. In other words, OSI is concerned with the interconnection aspects of cooperation (see Note) between systems, which is implied by the expression "systems interconnection".

The objective of OSI is to define a set of Recommendations to enable open systems to cooperate. A system which complies with the requirements of applicable OSI Recommendations in its cooperation with other systems is termed a real open system.

Note – Cooperation among open systems involves a broad range of activities of which the following have been identified:

a) interprocess communication, which concerns the exchange of information and the synchronization of activity between OSI application-processes;

b) data representation, which concerns all aspects of the creation and maintenance of data descriptions and data transformations for reformatting data exchanged between open systems;

c) data storage, which concerns storage media, and file and database systems for managing and providing access to data stored on the media;

d) process and resource management, which concerns the means by which OSI application-processes are declared, initiated and controlled, and the means by which they acquire OSI resources;

e) integrity and security, which concerns information processing constraints that must be preserved or assured during the operation of the open systems; and

f) program support, which concerns the definition, compilation, linking, testing, storage, transfer and access to the programs executed by OSI application-processes.

Some of these activities may imply exchange of information between the interconnected open systems and their interconnection aspects may, therefore, be of concern to OSI.

This Recommendation covers the elements of OSI aspects of these activities which are essential for early development of OSI Recommendations.

4.3 *Modelling the OSI environment*

The development of OSI Recommendations, i.e., Recommendations for the interconnection of real open systems, is assisted by the use of abstract models. To specify the external behaviour of interconnected real open systems, each real open system is replaced by a functionally equivalent abstract model of a real open system called an open system. Only the interconnection aspects of these open systems would strictly need to be described. However, to accomplish this, it is necessary to describe both the internal and external behaviour of these open systems. Only the external behaviour of open systems is retained as the standard of behaviour of real open systems. The description of the internal behaviour of open systems is provided in the Reference Model only to support the definition of the interconnection aspects. Any real system which behaves externally as an open system can be considered to be a real open system.

This abstract modelling is used in two steps.

First, basic elements of open systems and some key decisions concerning their organization and functioning, are developed. This constitutes the Reference Model of Open Systems Interconnection described in this Recommendation.

Then, the detailed and precise description of the functioning of the open system is developed in the framework formed by the Reference Model. This constitutes the services and protocols for Open Systems Interconnection which are the subject of other Recommendations.

It should be emphasized that the Reference Model does not, by itself, specify the detailed and precise functioning of the open system and, therefore, it does not *specify* the external behaviour of real open systems and does not imply the structure of the implementation of a real open system.

The reader not familiar with the technique of abstract modelling is cautioned that those concepts introduced in the description of open systems constitute an abstraction despite a similar appearance to concepts commonly found in real systems. Therefore real open systems need not be implemented as described by the Model.

Throughout the remainder of this Recommendation, only the aspects of real systems and application-processes which lie within the OSI environment are considered. Their interconnection is illustrated throughout this Recommendation as depicted in Figure 2/X.200.

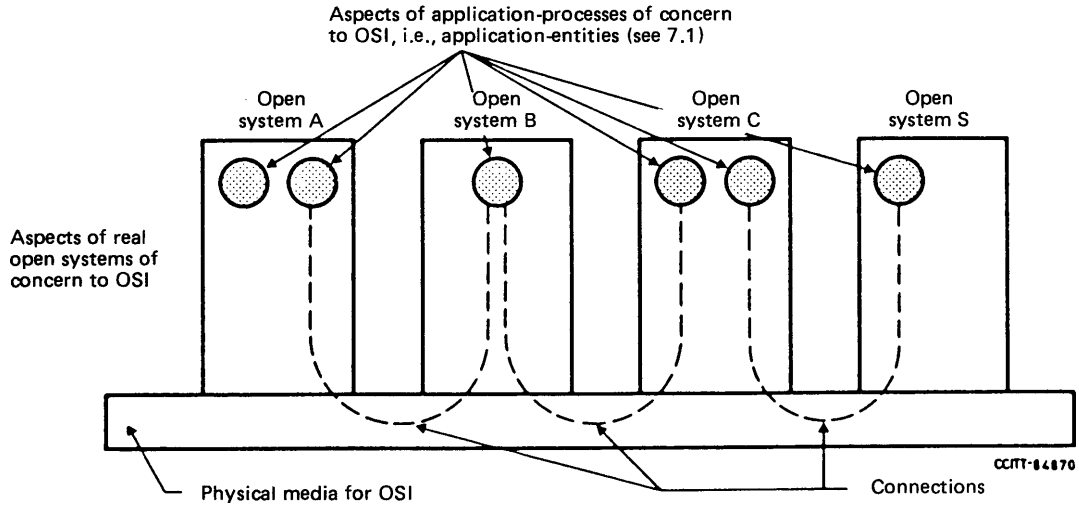


FIGURE 2/X.200

Basic elements of OSI

5 Concepts of a layered architecture

5.1 *Introduction*

Division 5 sets forth the architectural concepts that are applied in the development of the Reference Model of Open Systems Interconnection. Firstly, the concept of a layered architecture (with layers, entities, service-access-points, protocols, connections, etc.) is described. Secondly, identifiers are introduced for entities, service-access-points, and connections. Thirdly, service-access-points and data-units are described. Fourthly, elements of layer operation are described including connections, transmission of data, and error functions. Then, routing aspects are introduced and finally, management aspects are discussed.

The concepts described in division 5 are those required to describe the Reference Model of Open Systems Interconnection. However, not all of the concepts described are employed in each layer of the Reference Model.

Four elements are basic to the Reference Model (see Figure 2/X.200):

a) open systems;

b) the application-entities which exist within the Open Systems Interconnection environment;

c) the connections (see § 5.3) which join the application-entities and permit them to exchange information (see Note 1); and

d) the physical media for Open Systems Interconnection.

Note 1 – This Basic Reference Model of Open Systems Interconnection is based on the assumption that a connection is requir' for the transfer of data. An addendum to this Recommendation is currently being developed to extend .e description to cover the connectionless forms of data transmission which may be found in a wide variety of data communications techniques (e.g. local area networks, digital radio, etc.) and applications (e.g. remote sensing and banking).

Note 2 – Security aspects which are also general architectural elements of protocols are not discussed in this Recommendation.

5.2 *Principles of layering*

5.2.1 *Definitions*

5.2.1.1 **(N)-subsystem**

An element in a hierarchical division of an open system which interacts directly only with elements in the next higher division or the next lower division of that open system.

5.2.1.2 **(N)-layer**

A sub-division of the OSI architecture, constituted by subsystems of the same rank (N).

5.2.1.3 **(N)-entity**

An active element within an (N)-subsystem.

5.2.1.4 **peer-entities**

Entities within the same layer.

5.2.1.5 **sublayer**

A sub-division of a layer.

5.2.1.6 **(N)-service**

A capability of the (N)-layer and the layers beneath it, which is provided to (N + 1)-entities at the boundary between the (N)-layer and the (N + 1)-layer.

5.2.1.7 **(N)-facility**

A part of an (N)-service.

5.2.1.8 **(N)-function**

A part of the activity of (N)-entities.

5.2.1.9 **(N)-service-access-point**

The point at which (N)-services are provided by an (N)-entity to an (N + 1)-entity.

5.2.1.10 **(N)-protocol**

A set of rules and formats (semantic and syntactic) which determines the communication behaviour of (N)-entities in the performance of (N)-functions.

5.2.2 *Description*

The basic structuring technique in the Reference Model of Open Systems Interconnection is layering. According to this technique, each open system is viewed as logically composed of an ordered set of subsystems, represented for convenience in the vertical sequence shown in Figure 3/X.200. Adjacent subsystems communicate through their common boundary. Subsystems of the same rank (N) collectively for the (N)-layer of the Reference Model of Open Systems Interconnection. An (N)-subsystem consists of one or several (N)-entities. Entities exist in each layer. Entities in the same layer are termed peer-entities. Note that the highest layer does not have an (N + 1)-layer above it and the lowest layer does not have an (N − 1)-layer below it.

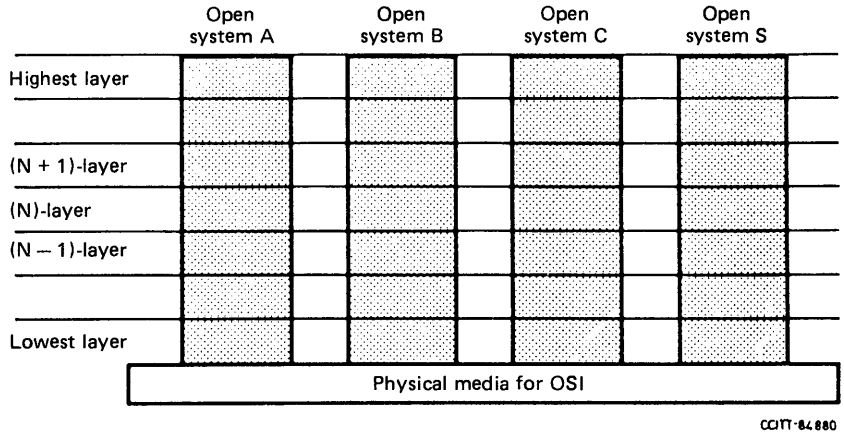


FIGURE 3/X.200

Layering in co-operating open systems

Not all peer (N)-entities need or even can communicate. There may be conditions which prevent this communication (e.g. they are not in interconnected open systems, or they do not support the same protocol subsets).

Note 1 – Type and Instance.

The distinction between the *type* of some object and an *instance* of that object is a distinction of significance for OSI. A type is a description of a class of objects. An instance of this type is any object that conforms to this description. The instances of the same type constitute a class. A type, and any instance of this type can be referred to by an individual name. Each nameable instance and the type to which this instance belongs carry distinguishable names.

For example, given that a programmer has written a computer program, that programmer has generated a type of something, where instances of that type are created every time that particular program is invoked into execution by a computer. Thus, a FORTRAN compiler is a type and each occasion where a copy of that program is invoked in a data processing machine displays an instance of that program.

Consider now an (N)-entity in the OSI context. It, too, has two aspects, a type and a collection of instances. The type of an (N)-entity is defined by the specific set of (N)-layer functions it is able to perform. An instance of that type of (N)-entity is a specific invocation of whatever it is within the relevant open system that provides the (N)-layer functions called for by its type for a particular occasion of communication. It follows from these observations that (N)-entity types refer only to the properties of an association between peer (N)-entities, while an (N)-entity instance refers to the specific, dynamic occasions of actual information exchange.

It is important to note that actual communication occurs only between (N)-entity instances at all layers. It is only at connection establishment time (or its logical equivalent during a recovery process) that (N)-entity types are explicitly relevant. Actual connections are always made to specific (N)-entity instances, although a request for a connection may well be made for arbitrary (N)-entity instances of a specified type. Nothing in Recommendation X.200, however, precludes the request for a connection with a specific (named) instance of a peer (N)-entity. If an (N)-entity instance is aware of the name of its peer (N)-entity instance, it is able to request another connection to that (N)-entity instance.

Note 2 – It may be necessary to further divide a layer into small substructures called sublayers and to extend the technique of layering to cover other dimensions of Open Systems Interconnection. A sublayer is described as a grouping of functions in a layer which may be bypassed. The bypassing of all the sublayers of a layer is not allowed. A sublayer uses the entities and connections of its layer. The detailed definition or additional characteristics of a sublayer are for further study.

Except for the highest layer, each (N)-layer provides (N + 1)-entities in the (N + 1)-layer with (N)-services. The highest layer is assumed to represent all possible uses of the services which are provided by the lower layers.

Note 1 – Not all open systems provide the initial source or final destination of data, such open systems need not contain the higher layers of the architecture (see Figures 6/X.200 and 13/X.200).

Note 2 – Classes of service may be defined within the (N)-services. The precise definition of the term "classes of service" is for further study.

Each service provided by an (N)-layer may be tailored by the selection of one or more (N)-facilities which determine the attributes of that service. When a single (N)-entity cannot by itself fully support a service requested by an (N + 1)-entity it calls upon the cooperation of other (N)-entities to help complete the service request. In order to cooperate, (N)-entities in any layer, other than those in the lowest layer, communicate by means of the set of services provided by the (N – 1)-layer (see Figure 4/X.200). The entities in the lowest layer are assumed to communicate directly via the physical media which connect them.

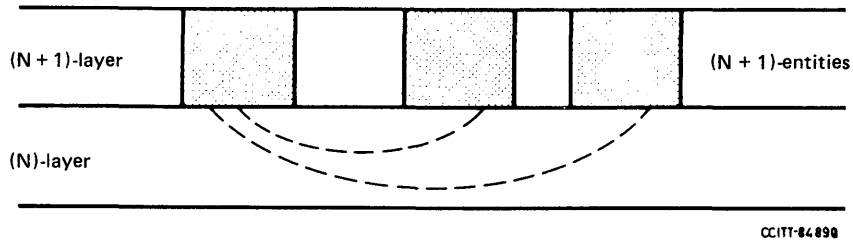


FIGURE 4/X.200

(N + 1)-entities in the (N + 1)-layer communicate through the (N)-layer

The services of an (N)-layer are provided to the (N + 1)-layer, using the (N)-functions performed within the (N)-layer and as necessary the services available from the (N − 1)-layer.

An (N)-entity may provide services to one or more (N + 1)-entities and use the services of one or more (N − 1)-entities. An (N)-service-access-point is the point at which a pair of entities in adjacent layers use or provide services (see Figure 7/X.200).

Cooperation between (N)-entities is governed by one or more (N)-protocols. The entities and protocols within a layer are illustrated in Figure 5/X.200.

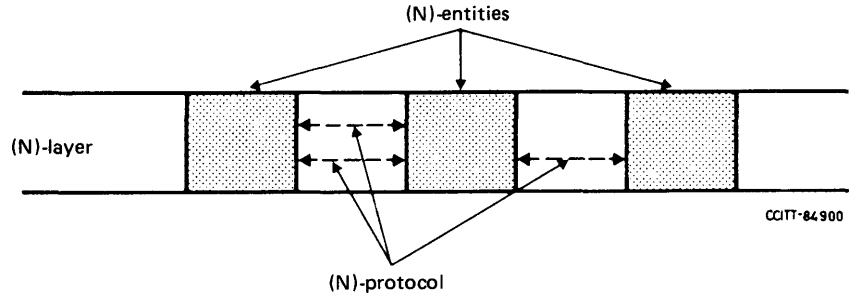


FIGURE 5/X.200

(N)-protocols between (N)-entities

5.3 *Communication between peer entities*

5.3.1 *Definitions*

5.3.1.1 **(N)-connection**

An association established by the (N)-layer between two or more (N + 1)-entities for the transfer of data.

5.3.1.2 **(N)-connection-endpoint**

A terminator at one end of an (N)-connection within an (N)-service-access-point.

5.3.1.3 **multi-endpoint-connection**

A connection with more than two connection-endpoints.

5.3.1.4 **correspondent (N)-entities**

(N)-entities with an (N − 1)-connection between them.

5.3.1.5 **(N)-relay**

An (N)-function by means of which an (N)-entity forwards data received from one correspondent (N)-entity to another correspondent (N)-entity.

5.3.1.6 **(N)-data-source**[2)]

An (N)-entity that sends (N − 1)-service-data-units (see § 5.6.1.7) on an (N − 1)-connection.

[2)] These definitions are not for use in this Recommendation but are for use in future OSI Recommendations.

5.3.1.7 **(N)-data-sink** [3]

An (N)-entity that receives (N − 1)-service-data-units on an (N − 1)-connection.

5.3.1.8 **(N)-data-transmission** [3]

An (N)-facility which conveys (N)-service-data-units from one (N + 1)-entity to one or more (N + 1)-entities.

5.3.1.9 **(N)-duplex-transmission** [3]

(N)-data transmission in both directions at the same time.

5.3.1.10 **(N)-half-duplex-transmission** [3]

(N)-data transmission in either direction one direction at a time; the choice of direction is controlled by an (N + 1)-entity.

5.3.1.11 **(N)-simplex-transmission** [3]

(N)-data-transmission in one pre-assigned direction.

5.3.1.12 **(N)-data-communication** [3]

An (N)-function which transfers (N)-protocol-data-units (see § 5.6.1.3) according to an (N)-protocol over one or more (N − 1)-connections.

5.3.1.13 **(N)-two-way-simultaneous-communication**

(N)-data-communication in both directions at the same time.

5.3.1.14 **(N)-two-way alternate communication**

(N)-data communication in both directions, one direction at a time.

5.3.1.15 **(N)-one-way communication**

(N)-data communication in one pre-assigned direction.

5.3.2 *Description*

For information to be exchanged between two or more (N + 1)-entities, an association is established between them in the (N)-layer using an (N)-protocol.

Note – Classes of protocols may be defined within the (N)-protocols. The precise definition of the term "classes of protocols" is for further study.

This association is called an (N)-connection. (N)-connections are provided by the (N)-layer between two or more (N)-service-access-points. The terminator of an (N)-connection at an (N)-service-access-point is called an (N)-connection-endpoint. A connection with more than two connection-endpoints is termed a multi-endpoint-connection. (N)-entities with a connection between them are termed correspondent (N)-entities.

(N + 1)-entities can communicate only by using the services of the (N)-layer. There are instances where services provided by the (N)-layer do not permit direct access between all of the (N + 1)-entities which have to communicate. If this is the case, communication can still occur if some other (N + 1)-entity can act as a relay between them (see Figure 6/X.200).

The fact that communication is relayed by a chain of (N + 1)-entities is known neither by the (N)-layer nor by the (N + 2)-layer.

---

[3] These definitions are not for use in this Recommendation but are for use in future OSI Recommendations.

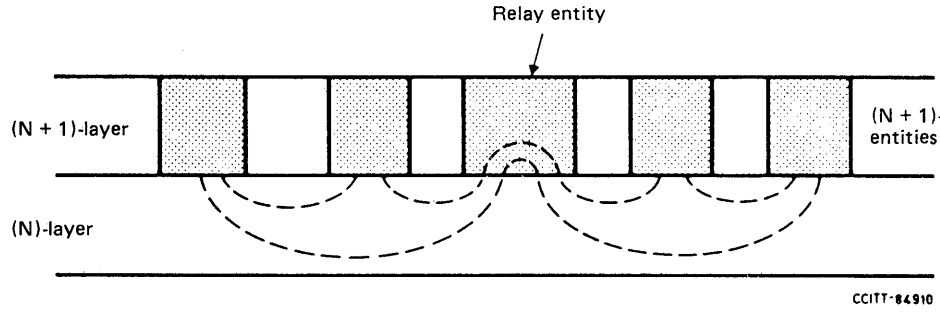


FIGURE 6/X.200

Communication through a relay

5.4 *Identifiers*

5.4.1 *Definitions*

5.4.1.1 **title**

A permanent identifier for an entity.

5.4.1.2 **title-domain**

A subset of the title space of the OSI environment.

5.4.1.3 **title-domain-name**

An identifier which uniquely identifies a title-domain with the OSI environment.

Note – Title-domains of primary importance are the layers. In this specific case, the title-domain-name identifies the (N)-layer.

5.4.1.4 **local-title**

A title which is unique within a title-domain.

5.4.1.5 **global-title**

A title which is unique within the OSI environment and comprises two parts, a title-domain-name and a local-title.

5.4.1.6 **(N)-address; (N)-service-access-point-address**

An identifier which tells where an (N)-service-access-point may be found.

5.4.1.7 **(N)-directory**

An (N)-function by which the global title of an (N)-entity is translated into the (N – 1)-address of an (N – 1)-service-access-point to which the (N)-entity is attached.

5.4.1.8 **(N)-address-mapping**

An (N)-function which provides the mapping between the (N)-addresses and the (N – 1)-addresses associated with an (N)-entity.

5.4.1.9 **routing**

A function within a layer which translates the title of an entity or the service-access-point-address to which the entity is attached into a path by which the entity can be reached.

5.4.1.10 **(N)-connection-endpoint-identifier**

An identifier of an (N)-connection-endpoint which can be used to identify the corresponding (N)-connection at an (N)-service-access-point.

5.4.1.11 **(N)-connection-endpoint-suffix**

A part of an (N)-connection-endpoint-identifier which is unique within the scope of an (N)-service-access-point.

5.4.1.12 **multi-connection-endpoint-identifier**

An identifier which specifies the connection-endpoint of a multi-endpoint-connection which should accept the data that is being transferred.

5.4.1.13 **(N)-service-connection-identifier**

An identifier which uniquely specifies an (N)-connection within the environment of the correspondent (N + 1)-entities.

5.4.1.14 **(N)-protocol-connection-identifier**

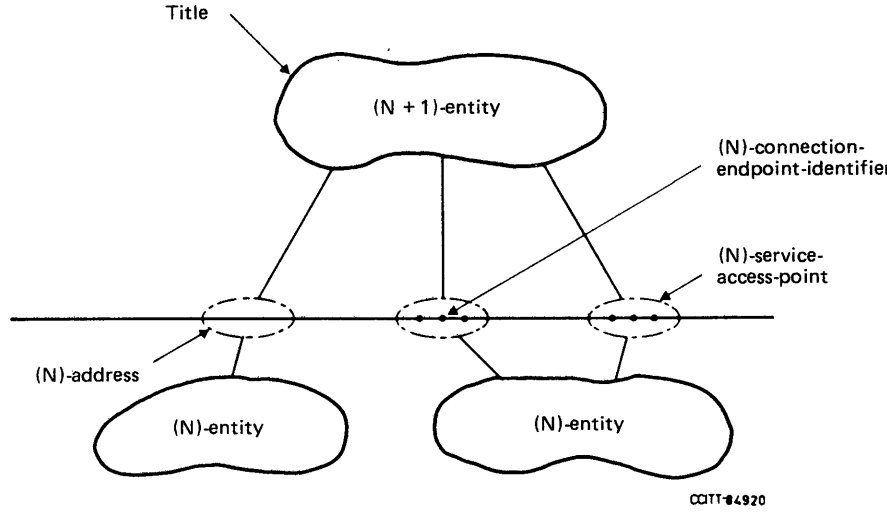
An identifier which uniquely specifies an individual (N)-connection within the environment of the multiplexed (N − 1)-connection.

5.4.1.15 **(N)-suffix**

A part of an (N)-address which is unique within the (N)-service-access-point.

5.4.2 *Description*

An (N)-service-access-point-address, or (N)-address for short, identifies a particular (N)-service-access-point to which an (N + 1)-entity is attached (see Figure 7/X.200). When the (N + 1)-entity is detached from the (N)-service-access-point, the (N)-address no longer provides access to the (N + 1)-entity. If the (N)-service-access-point is reattached to a different (N + 1)-entity, then the (N)-address identifies the new (N + 1)-entity and not the old one.



Note – Dashed arrows refer to identifiers

FIGURE 7/X.200

Entities, service-access-points and identifiers

The use of an (N)-address to identify an (N + 1)-entity is the most efficient mechanism if the permanence of the attachment between the (N + 1)-entity and the (N)-service-access-point can be assured. If there is a requirement to identify an (N + 1)-entity regardless of its current location, then the global-title assures correct identification.

An (N)-directory is an (N)-function which translates global-titles of peer (N)-entities into the (N – 1)-addresses through which they cooperate.

Interpretation of the correspondence between the (N)-addresses served by an (N)-entity and the (N – 1)-addresses used for accessing (N – 1)-services is performed by an (N)-address-mapping function.

Two particular kinds of (N)-address-mapping functions may exist within a layer:

a) hierarchical (N)-address-mapping; and

b) (N)-address-mapping by tables.

If an (N)-address is always mapped into only one (N – 1)-address then hierarchical construction of addresses can be used (see Figure 8/X.200). The (N)-address-mapping function need only recognize the hierarchical structure of an (N)-address and extract the (N – 1)-address it contains.

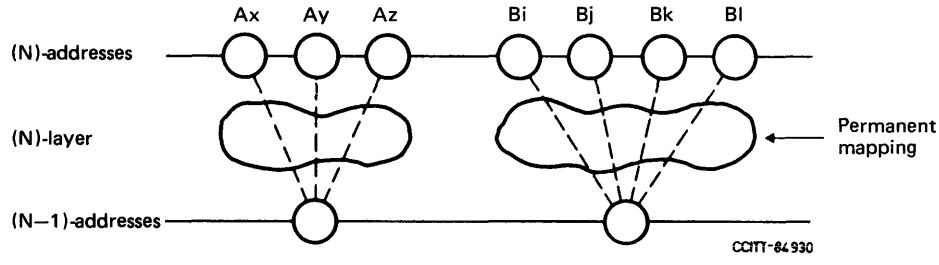


FIGURE 8/X.200

Hierarchical (N)-address-mapping

In this case, an (N)-address consists of two parts:

a) an (N – 1)-address of the (N)-entity which is supporting the current (N)-service-access-point of the (N + 1)-entity;

b) an (N)-suffix which makes the (N)-service-access-point uniquely identifiable within the scope of the (N – 1)-address.

Within a given layer, a hierarchical structure of addresses simplifies (N)-address-mapping functions because of the permanent nature of the mapping it presupposes. It is not imposed by the model in all layers in order to allow more flexibility in (N)-address-mappings and to cover the case where an (N)-entity attached to more than one (N – 1)-service-access-point supports only one (N)-service-access-point.

If the previous condition is not true, i.e. either an (N)-address can be mapped into several (N – 1)-addresses, or an (N)-address is not permanently mapped into the same (N – 1)-address, then hierarchical construction of an address is not possible and the (N)-address-mapping function may use tables to translate (N)-addresses into (N – 1)-addresses.

The structure of an (N)-address is known by the (N)-entity which is attached to the identified (N)-service-access-point. However, the (N + 1)-entity does not know this structure.

If an (N + 1)-entity has two or more (N)-service-access-points with either the same (N)-entity or different (N)-entities, the (N)-entities have no knowledge of this fact. Each (N)-service-access-point is considered to identify a different (N + 1)-entity from the perspective of the (N)-entities.

A routing function translates the (N)-address of an (N + 1)-entity into a path or route by which the (N + 1)-entity may be reached.

An (N + 1)-entity may establish an (N)-connection with another (N + 1)-entity by using an (N)-service. When an (N + 1)-entity establishes an (N)-connection with another (N + 1)-entity, each (N + 1)-entity is given an (N)-connection-endpoint-identifier by its supporting (N)-entity. The (N + 1)-entity can then distinguish the new connection from all other (N)-connections accessible at the (N)-service-access-point it is using. This (N)-connection-endpoint-identifier is required to be unique within the scope of the (N + 1)-entity which will use the (N)-connection.

The (N)-connection-endpoint-identifier consists of two parts:

a) the (N)-address of the (N)-service-access-point which will be used in conjunction with the (N)-connection; and

b) an (N)-connection-endpoint-suffix which is unique within the scope of the (N)-service-access-point.

A multi-endpoint-connection requires multi-connection-endpoint-identifiers. Each such identifier is used to specify which connection-endpoint should accept the data which is being transferred. A multi-connection-endpoint-identifier must be unique within the scope of the connection within which it is used.

The (N)-layer may provide to the (N + 1)-entities an (N)-service-connection-identifier which uniquely specifies the (N)-connection within the environment of the correspondent (N + 1)-entities.

5.5 *Properties of service-access-points*

An (N + 1)-entity requests (N)-services via an (N)-service-access-point which permits the (N + 1)-entity to interact with an (N)-entity.

Both the (N)- and (N + 1)-entities attached to an (N)-service-access-point are in the same system.

An (N + 1)-entity may concurrently be attached to one or more (N)-service-access-points attached to the same or different (N)-entities.

An (N)-entity may concurrently be attached to one or more (N + 1)-entities through (N)-service-access-points.

An (N)-service-access-point is attached to only one (N)-entity and to only one (N + 1)-entity at a time.

An (N)-service-access-point may be detached from an (N + 1)-entity and reattached to the same or another (N + 1)-entity.

An (N)-service-access-point may be detached from an (N)-entity and reattached to the same or another (N)-entity.

An (N)-service-access-point is located by means of its (N)-address. An (N)-address is used by an (N + 1)-entity to request an (N)-connection.

5.6 *Data Units*

5.6.1 *Definitions*

5.6.1.1 **(N)-protocol-control-information**

Information exchanged between (N)-entities, using an (N − 1)-connection, to coordinate their joint operation.

5.6.1.2 **(N)-user-data**

The data transferred between (N)-entities on behalf of the (N + 1)-entities for whom the (N)-entities are providing services.

5.6.1.3 **(N)-protocol-data-unit**

A unit of data specified in an (N)-protocol and consisting of (N)-protocol-control-information and possibly (N)-user-data.

5.6.1.4 **(N)-interface-control-information**

Information transferred between an (N + 1)-entity and an (N)-entity to coordinate their joint operation.

5.6.1.5 **(N)-interface-data**

Information transferred from an (N + 1)-entity to an (N)-entity for transmission to a correspondent (N + 1)-entity over an (N)-connection, or conversely, information transferred from an (N)-entity to an (N + 1)-entity after being received over an (N)-connection from a correspondent (N + 1)-entity.

5.6.1.6 **(N)-interface-data-unit**

The unit of information transferred across the (N)-service-access-point between an (N + 1)-entity and an (N)-entity in a single interaction. Each (N)-interface-data-unit contains (N)-interface-control-information and may also contain the whole or part of an (N)-service-data-unit.

5.6.1.7 **(N)-service-data-unit**

An amount of (N)-interface-data whose identity is preserved from one end of an (N)-connection to the other.

5.6.1.8 **expedited (N)-service-data-unit, (N)-expedited-data-unit**

A small (N)-service-data-unit whose transfer is expedited. The (N)-layer ensures that an expedited-data-unit will not be delivered after any subsequent service-data-unit or expedited unit sent on that connection.

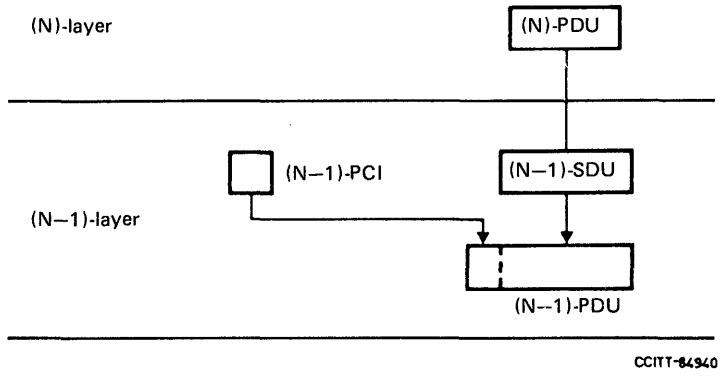
5.6.2 *Description*

Information is transferred in various types of data units between peer-entities and between entities attached to a specific service-access-point. The data-units are defined in sub-division 5.6.1 and the relationships among them are illustrated in Figures 9/X.200 and 10/X.200.

	Control	Data	Combined
(N)-(N) peer-entities	(N)-protocol-control-information	(N)-user-data	(N)-protocol-data-units
(N + 1)-(N) adjacent layers	(N)-interface-control-information	(N)-interface-data	(N)-interface-data-unit

FIGURE 9/X.200

Relationships among data-units



PCI = protocol-control-information
PDU = protocol-data-unit
SDU = service-data-unit

Note 1 – This figure assumes that neither segmenting nor blocking of (N)-service-data-units is performed (see § 5.7.6.5).

Note 2 – This figure does not imply any positional relationship between protocol-control-information and user-data in protocol-data-units.

Note 3 – An (N)-protocol-data-unit may be mapped one-to-one into an (N−1)-service-data-unit, but other relationships are possible (see Figure 11/X.200).

FIGURE 10/X.200

An illustration of mapping between data units in adjacent layers

Except for the relationships defined in Figures 9/X.200 and 10/X.200, there is no overall architectural limit to the size of data units. There may be other size limitations at specific layers.

The size of (N)-interface-data-units is not necessarily the same at each end of the connection.

Data may be held within a connection until a complete service-data-unit is put into the connection.

5.7 *Elements of layer operation*

5.7.1 *Definitions*

5.7.1.1 **(N)-protocol-identifier**

An identifier used between correspondent (N)-entities to select a specific (N)-protocol to be used on a particular (N – 1)-connection.

5.7.1.2 **centralized multi-endpoint-connection**

A multi-endpoint-connection where data sent by the entity associated with the central connection-endpoint is received by all other entities, while data sent by one of the other entities is only received by the central entity.

5.7.1.3 **decentralized multi-endpoint-connection**

A multi-endpoint-connection such that data sent by an entity associated with a connection-endpoint is received by all other entities.

5.7.1.4 **multiplexing**

A function within the (N)-layer by which one (N – 1)-connection is used to support more than one (N)-connection.

Note – The term multiplexing is also used in a more restricted sense to refer to the function performed by the sending (N)-entity while the term demultiplexing is used to refer to the function performed by the receiving (N)-entity.

5.7.1.5 **demultiplexing**

The function performed by an (N)-entity which identifies (N)-protocol-data-units for more than one (N)-connection within (N – 1)-service-data-units received on a single (N – 1)-connection. It is the reverse function of the multiplexing function performed by the (N)-entity sending the (N – 1)-service-data-units.

5.7.1.6 **splitting**

A function within the (N)-layer by which more than one (N – 1)-connection is used to support one (N)-connection.

Note – The term splitting is also used in a more restrictive sense to refer to the function performed by the sending (N)-entity while the term recombining is used to refer to the function performed by the receiving (N)-entity.

5.7.1.7 **recombining**

The function performed by an (N)-entity which identifies (N)-protocol-data-units for a single (N)-connection in (N – 1)-service-data-units received on more than one (N – 1)-connection. It is the reverse function of the splitting function performed by the (N)-entity sending the (N – 1)-service-data-units.

5.7.1.8 **flow control**

A function which controls the flow of data within a layer or between adjacent layers.

5.7.1.9 **segmenting**

A function performed by an (N)-entity to map one (N)-service-data-unit into multiple (N)-protocol-data-units.

5.7.1.10 **reassembling**

A function performed by an (N)-entity to map multiple (N)-protocol-data-units into one (N)-service-data-unit. It is the reverse function of segmenting.

5.7.1.11 **blocking**

A function performed by an (N)-entity to map multiple (N)-service-data-units into one (N)-protocol-data-unit.

5.7.1.12 **deblocking**

A function performed by an (N)-entity to identify multiple (N)-service-data-units which are contained in one (N)-protocol-data-unit. It is the reverse function of blocking.

5.7.1.13 **concatenation**

A function performed by an (N)-entity to map multiple (N)-protocol-data-units into one (N – 1)-service-data-unit.

5.7.1.14 **separation**

A function performed by an (N)-entity to identify multiple (N)-protocol-data-units which are contained in one (N – 1)-service-data-unit. It is the reverse function of concatenation.

5.7.1.15 **sequencing**

A function performed by the (N)-layer to preserve the order of (N)-service-data-units that were submitted to the (N)-layer.

5.7.1.16 **acknowledgement**

A function of the (N)-layer which allows a receiving (N)-entity to inform a sending (N)-entity of the receipt of an (N)-protocol-data-unit.

5.7.1.17 **reset**

A function which sets the correspondent (N)-entities to a predefined state with a possible loss or duplication of data.

Note – Blocking and concatenation, though close to each other (they both permit grouping of data-units) are different (see definitions §§ 5.7.1.11 and 5.7.1.13). These two functions may serve different purposes. For instance, concatenation permits the (N)-layer to group one or several acknowledgement (N)-PDUs with one (or several) (N)-PDUs containing user data, and this would not be possible with the blocking function only. Note also that the two functions may be combined so that the (N)-layer performs blocking and concatenation.

5.7.2 *Protocol selection*

One or more (N)-protocols may be defined for the (N)-layer. An (N)-entity may employ one or more (N)-protocols.

Meaningful communication between (N)-entities over an (N – 1)-connection requires the agreed selection of one (N)-protocol. (N)-protocol-identifiers name the specific protocols defined.

5.7.3 *Properties of connections*

An (N)-connection is an association established for communication between two or more (N + 1)-entities, identified by their (N)-addresses. An (N)-connection is offered as a service by the (N)-layer, so that information may be exchanged between the (N + 1)-entities.

An (N + 1)-entity may have, simultaneously, one or more (N)-connections with other (N + 1)-entities, with any given (N + 1)-entity, and with itself.

An (N)-connection is established by referencing, either explicitly or implicitly, an (N)-address for the source (N + 1)-entity and an (N)-address for each of one or more destination (N + 1)-entities.

The source (N)-address and one or more of the destination (N)-addresses may be the same. One or more of the destination (N)-addresses may be the same while the source (N)-address is different. All may be different.

One (N)-connection-endpoint is constructed for each (N)-address referenced explicitly or implicitly when an (N)-connection is established.

An (N + 1)-entity accesses an (N)-connection via an (N)-service-access-point.

An (N)-connection has two or more (N)-connection-endpoints.

An (N)-connection-endpoint is not shared by (N + 1)-entities or (N)-connections.

An (N)-connection-endpoint relates three elements:

a) an (N + 1)-entity;

b) an (N)-entity; and

c) an (N)-connection.

The (N)-entity and the (N + 1)-entity related by an (N)-connection-endpoint are those implied by the (N)-address referenced when the (N)-connection is established.

An (N)-connection-endpoint has an identifier, called an (N)-connection-endpoint-identifier, which is unique within the scope of the (N + 1)-entity which is bound to the (N)-connection-endpoint.

An (N)-connection-endpoint-identifier is not the same as an (N)-address.

An (N + 1)-entity references an (N)-connection using its (N)-connection-endpoint-identifier.

Multi-endpoint-connections are connections which have three or more connection-endpoints. Two types of multi-endpoint-connection are defined.[4)]

a) centralized; and

b) decentralized.

A centralized multi-endpoint-connection has a central connection-endpoint. Data sent by the entity associated with the central connection-endpoint is received by the entities associated with all other connection-endpoints. The data sent by an entity associated with any other connection-endpoint is received only by the entity associated with the central connection-endpoint.

On a decentralized multi-endpoint-connection, data sent by an entity associated with any connection-endpoint is received by the entities associated with all of the other connection-endpoints.

5.7.4 *Connection establishment and release*

The establishment of an (N)-connection by peer-entities of an (N)-layer requires the following:

a) the availability of an (N − 1)-connection between the supporting (N)-entities; and

b) both (N)-entities be in a state in which they can execute the connection establishment protocol exchange.

If it is not already available, an (N − 1)-connection has to be established by peer-entities of the (N − 1)-layer. This requires, for the (N − 1)-layer, the same conditions as described above for the (N)-layer.

The same consideration applies downwards until either an available connection or the physical medium for Open Systems Interconnection is encountered.

Depending upon the characteristics of the (N − 1)-service and of the establishment protocol exchange, the establishment of an (N)-connection may or may not be done in conjunction with the establishment of the (N − 1)-connection.

The characteristics of the (N)-service with regard to the establishment of the (N)-connection vary depending upon whether or not (N)-user-data can be transferred by the connection establishment protocol exchange for each direction of the (N)-connection.

Where (N)-user-data is transferred by the (N)-connection establishment protocol exchange, the (N + 1)-protocol may take advantage of this to allow an (N + 1)-connection to be established in conjunction with the establishment of the (N)-connection.

The release of an (N)-connection is normally initiated by one of the (N + 1)-entities associated in it.

The release of an (N)-connection may also be initiated by one of the (N)-entities supporting it as a result of an exception condition occurring in the (N)-layer or the layers below.

Depending upon the conditions, release of an (N)-connection may result in the discarding of (N)-user-data.

The orderly release of an (N)-connection requires either the availability of an (N − 1)-connection, or a common reference to time (e.g. time of failure of the (N − 1)-connection and common time-out). In addition, it requires that both (N)-entities are in a state in which they can execute the connection release protocol exchange. It is important to note, however, that the release of an (N − 1)-connection does not necessarily cause the release of the (N)-connection(s) which were using it; the (N − 1)-connection can be re-established, or another (N − 1)-connection substituted.

The characteristics of the (N)-service with regard to the release of an (N)-connection can be of two kinds:

a) (N)-connections are either released immediately when the release protocol exchange is initiated ((N)-user-data not yet delivered may be discarded); or

b) release is delayed until all (N)-user-data sent previous to the initiation of the release protocol exchange has been delivered (i.e., delivery confirmation has been received.)

(N)-user-data may be transferred by the connection release protocol exchange.

Some (N)-protocols may provide for the combining of connection establishment and connection release protocol exchanges.

[4)] Other types of multi-endpoint-connections are for further study.

5.7.5 *Multiplexing and splitting*

Within the (N)-layer, (N)-connections are mapped onto (N − 1)-connections. The mapping may be one of three kinds:

a) one-to-one;

b) many (N)-connections to one (N − 1)-connection (multiplexing); and

c) one (N)-connection to many (N − 1)-connections (splitting).

Multiplexing may be needed in order to:

a) make more efficient or more economic use of the (N − 1)-service; and

b) provide several (N)-connections in an environment where only a single (N − 1)-connection exists.

Splitting may be needed in order to:

a) improve reliability where more than one (N − 1)-connection is available;

b) provide the required grade of performance, through the utilization of multiple (N − 1)-connections; and

c) obtain cost benefits by the utilization of multiple low cost (N − 1)-connections each with less than the required grade of performance.

Multiplexing and splitting each involve a number of associated functions which may not be needed for one-to-one connection mapping.

The functions associated with multiplexing are:

a) identification of the (N)-connection for each (N)-protocol-data-unit transferred over the (N − 1)-connection, in order to ensure that (N)-user-data from the various multiplexed (N)-connections are not mixed. This identification is distinct from that of the (N)-connection-endpoint-identifiers and is called an (N)-protocol-connection-identifier;

b) flow control on each (N)-connection in order to share the capacity of the (N − 1)-connection (see § 5.7.6.4); and

c) scheduling the next (N)-connection to be serviced over the (N − 1)-connection when more than one (N)-connection is prepared to send data.

The functions associated with splitting are:

a) scheduling the utilization of multiple (N − 1)-connections used in splitting a single (N)-connection; and

b) resequencing of (N)-protocol-data-units associated with an (N)-connection since they may arrive out of sequence even when each (N − 1)-connection guarantees sequence of delivery (see § 5.7.6.6).

5.7.6 *Transfer of data*

5.7.6.1 *Normal Data transfer*

Control information and user data are transferred between (N)-entities in (N)-protocol-data-units. An (N)-protocol-data-unit is a unit of data specified in an (N)-protocol and contains (N)-protocol-control-information and possibly (N)-user-data.

(N)-protocol-control-information is transferred between (N)-entities using the (N − 1)-connection. (N)-protocol-control-information is any information that supports the joint operation of (N)-entities. (N)-user-data is passed transparently between (N)-entities over an (N − 1)-connection.

An (N)-protocol-data-unit has an arbitrary, but finite, size. (N)-protocol-data-units are mapped into (N − 1)-service-data-units. The interpretation of an (N)-protocol-data-unit is defined by the (N)-protocol in effect for the (N − 1)-connection.

An (N)-service-data-unit is transferred between an (N + 1)-entity and an (N)-entity, through an (N)-service-access-point, in the form of one or more (N)-interface-data-units. The (N)-service-data-unit is transferred as (N)-user-data in one or more (N)-protocol-data-units.

The exchange of data under the rules of an (N)-protocol can only occur if an (N − 1)-connection exists. If an (N − 1)-connection does not exist, it must be established before an exchange of data can occur (see § 5.7.4).

5.7.6.2 *Data transfer during connection establishment and release*

(N)-user-data may be transferred in the (N)-connection establishment protocol exchange and in the (N)-connection release protocol exchange.

The connection release protocol exchange may be combined with the connection establishment protocol exchange (see § 5.7.4) to provide means for the delivery of a single unit of (N)-user-data between correspondent (N + 1)-entities with a confirmation of receipt.

5.7.6.3 *Expedited transfer of data*

An expedited-data-unit is a service-data-unit which is transferred and/or processed with priority over normal service-data-units. An expedited data transfer service may be used for signalling and interrupt purposes.

Expedited data flow is independent of the states and operation of the normal data flow, although the data sent on the two flows may be logically related. Conceptually, a connection that supports expedited flow can be viewed as having two subchannels, one for normal data, the other for expedited data. Data sent on the expedited channel is assumed to be given priority over normal data.

The transfer guarantees that an expedited-data-unit will not be delivered after any subsequent normal service-data-unit or expedited-data-unit sent on the connection.

Because the expedited flow is assumed to be used to transfer small amounts of data infrequently, simplified flow control mechanisms may be used on this data flow.

An expedited (N)-service-data-unit is intended to be processed by the receiving (N + 1)-entity with priority over normal (N)-service-data-units.

5.7.6.4 *Flow control*

If flow control functions are provided, they can operate only on protocol-data-units and interface-data-units.

Two types of flow control are identified:

a) peer flow control which regulates the rate at which (N)-protocol-data-units are sent to the peer (N)-entity on the (N)-connection. Peer flow control requires protocol definitions and is based on protocol-data-unit size; and

b) (N)-interface flow control which regulates the rate at which (N)-interface-data are passed between an (N + 1)-entity and an (N)-entity. (N)-interface flow control is based on the (N)-interface-data-unit size.

Multiplexing in a layer may require a peer flow control function for individual flows (see § 5.7.5).

Peer flow control functions require that flow control information be included in the (N)-protocol-control-information of an (N)-protocol-data-unit.

If the size of service-data-units exceeds the maximum size of the (N)-user-data portion of an (N)-protocol-data-unit, then first segmentation must be performed on the (N)-service-data-unit to make it fit within the (N)-protocol-data-units. Peer flow control can then be applied on the (N)-protocol-data-units.

5.7.6.5 *Segmenting, blocking and concatenation*

Data units in the various layers are not necessarily of compatible size. It may be necessary to perform segmenting, i.e., to map an (N)-service-data-unit into more than one (N)-protocol-data-unit. Similarly, segmenting may occur when (N)-protocol-data-units are mapped into (N − 1)-interface-data-units. Since it is necessary to preserve the identity of (N)-service-data-units on an (N)-connection, functions shall be available to identify the segments of an (N)-service-data-unit, and to allow the correspondent (N)-entities to reassemble the (N)-service-data-unit.

Segmenting may require that information be included in the (N)-protocol-control-information of an (N)-protocol-data-unit. Within a layer, (N)-protocol-control-information is added to an (N)-service-data-unit to form an (N)-protocol-data-unit when no segmenting or blocking is performed (see Figure 11a/X.200). If segmenting is performed, an (N)-service-data-unit is mapped into several (N)-protocol-data-units with added (N)-protocol-control-information (see Figure 11b/X.200).

Conversely, it may be necessary to perform blocking whereby several (N)-service-data-units with added (N)-protocol-control-information form an (N)-protocol-data-unit (see Figure 11c/X.200).

The Reference Model also permits concatenation whereby several (N)-protocol-data-units are concatenated into a single (N – 1)-service-data-unit (see Figure 11d/X.200).

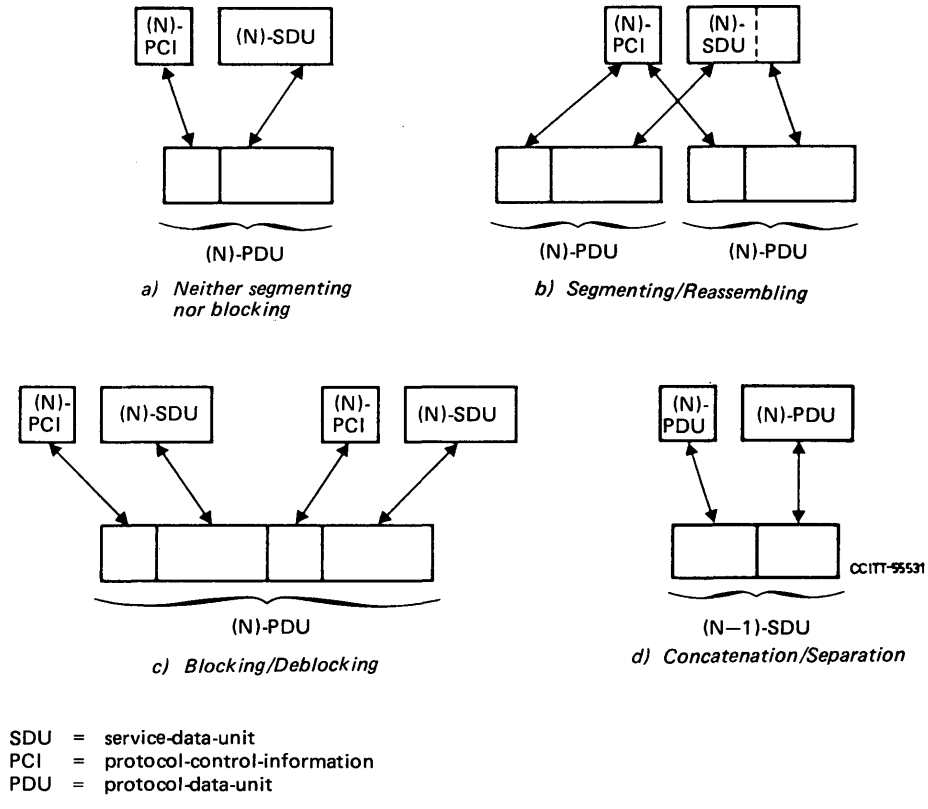
5.7.6.6 *Sequencing*

The (N – 1)-services provided by the (N – 1)-layer of the OSI architecture may not guarantee delivery of (N – 1)-service-data-units in the same order as it was submitted by the (N)-layer. If the (N)-layer needs to preserve the order of (N – 1)-service-data-units transferred through the (N – 1)-layer, sequencing mechanisms must be present in the (N)-layer. Sequencing may require additional (N)-protocol-control-information.

5.7.7 *Error functions*

5.7.7.1 *Acknowledgement*

An acknowledgement function may be used by peer (N)-entities using an (N)-protocol to obtain a higher probability of detecting protocol-data-unit loss than is provided by the (N – 1)-layer. Each (N)-protocol-data-unit transferred between correspondent (N)-entities is made uniquely identifiable, so that the receiver can inform the sender of the receipt of the (N)-protocol-data-unit. An acknowledgement function is also able to infer the non-receipt of (N)-protocol-data-units and take appropriate remedial action.



Note 1 – This figure does not imply any positional relationship between protocol-control-information and user-data in protocol-data-units.

Note 2 – In the case of concatenation, (N)-protocol-data-unit does not necessarily include an (N)-service-data-unit.

FIGURE 11/X.200

Relationship between (N)-service-data-unit, (N)-protocol-data-unit and (N–1)-service-data-unit within a layer

An acknowledgement function may require that information be included in the (N)-protocol-control-information of (N)-protocol-data-units.

The scheme for uniquely identifying (N)-protocol-data-units may also be used to support other functions such as detection of duplicate data-units, segmenting and sequencing.

Note – Other forms of acknowledgement such as confirmation of delivery and confirmation of performance of an action are for further study.

5.7.7.2 *Error detection and notification*

Error detection and notification functions may be used by an (N)-protocol to provide a higher probability of both protocol-data-unit error detection and data corruption detection than is provided by the (N – 1)-service.

Error detection and notification may require that additional information be included in the (N)-protocol-control-information of the (N)-protocol-data-unit.

5.7.7.3 *Reset*

Some services require a reset function to recover from a loss of synchronization between correspondent (N)-entities. A reset function sets the correspondent (N)-entities to a predefined state with a possible loss or duplication of data.

Note – Additional functions may be required to determine at what point reliable data transfer was interrupted.

A quantity of (N)-user-data may be conveyed in association with the (N)-reset function.

The reset function may require that information be included in the (N)-protocol-control-information of the (N)-protocol-data-unit.

5.8 *Routing*

A routing function within the (N)-layer enables communication to be relayed by a chain of (N)-entities. The fact that communication is being routed by intermediate (N)-entities is known by neither the lower layers nor by the higher layers. An (N)-entity which participates in a routing function may have a routing table.

5.9 *Management aspects of OSI*

5.9.1 *Definitions*

5.9.1.1 **application-management**

Functions in the Application Layer (see § 6.1) related to the management of OSI application-processes.

5.9.1.2 **application-management-application-entity**

An application-entity which executes application-management functions.

5.9.1.3 **OSI resources**

Data processing and data communication resources which are of concern to OSI.

5.9.1.4 **systems-management**

Functions in the Application Layer related to the management of various OSI resources and their status across all layers of the OSI architecture.

5.9.1.5 **systems-management-application-entity**

An application-entity which executes systems-management functions.

5.9.1.6 **layer-management**

Functions related to the management of the (N)-layer partly performed in the (N)-layer itself according to the (N)-protocol of the layer (activities such as activation and error control) and partly performed as a subset of systems-management.

5.9.2 *Introduction*

Within the OSI architecture there is a need to recognize the special problems of initiating, terminating, and monitoring activities and assisting in their harmonious operations, as well as handling abnormal conditions. These have been collectively considered as the management aspects of the OSI architecture. These concepts are essential to the operation of the interconnected open systems and therefore are included in the comprehensive description of the Reference Model described in subsequent divisions of this Recommendation.

The management activities which are of concern are those which imply actual exchanges of information between open systems. Only the protocols needed to conduct such exchanges are candidates for standardization within the OSI architecture.

This sub-division describes key concepts relevant to the management aspects, including the different categories of management activities and the positioning of such activities within the OSI architecture.

5.9.3 *Categories of management activities*

Only those management activities which imply actual exchanges of information between remote management entities are pertinent to the OSI architecture. Other management activities local to particular open systems are outside its scope.

Similarly, not all resources are pertinent to OSI. This Recommendation considers only OSI resources, i.e., those data processing and data communication resources which are of concern to OSI.

The following categories of management activities are identified:

a) application-management;

b) systems-management; and

c) layer-management.

5.9.3.1 *Application-management*

Application-management relates to the management of OSI application-processes. The following list is typical of activities which fall into this category but it is not exhaustive:

a) initialization of parameters representing application-processes;

b) initiation, maintenance and termination of application-processes;

c) allocation and de-allocation of OSI resources to application-processes;

d) detection and prevention of OSI resource interference and deadlock;

e) integrity and commitment control;

f) security control; and

g) checkpointing and recovery control.

The protocols for application-management reside within the Application Layer, and are handled by application-management-application-entities.

5.9.3.2 *Systems-management*

Systems-management relates to the management of OSI resources and their status across all layers of the OSI architecture. The following list is typical of activities which fall into this category but it is not exhaustive:

a) activation/deactivation management which includes:

   1) activation, maintenance and termination of OSI resources distributed in open systems, including physical media for OSI;

   2) some program loading functions;

   3) establishment/maintenance/release of connections between management entities; and

   4) open systems parameter initialization/modification;

b) monitoring which includes:
   1) reporting status or status changes; and
   2) reporting statistics; and

c) error control which includes:
   1) error detection and some of the diagnostic functions; and
   2) reconfiguration and restart.

The protocols for systems-management reside in the Application Layer, and are handled by systems-management-application-entities.

5.9.3.3 *Layer-management*

There are two aspects of layer-management. One of these is concerned with layer activities such as activation and error control. This aspect is implemented by the layer protocol to which it applies.

The other aspect of layer-management is a subset of systems-management. The protocols for these activities reside within the Application Layer and are handled by system-management-application-entities.

5.9.4 *Principles for positioning management functions*

Several principles are important in positioning management functions in the Reference Model of Open Systems Interconnection. They include the following:

a) both centralization and decentralization of management functions are allowed. Thus, the OSI architecture does not dictate any particular fashion or degree of centralization of such functions. This principle calls for a structure in which each open system is allowed to include any (subset of) systems-management functions and each subsystem is allowed to include any (subset of) layer-management functions; and

b) if it is necessary, connections between management entities are established when an open system which has been operating in isolation from other open systems, becomes part of the OSI environment.

Note – Other principles are for further study.

6 Introduction

6.1 *Specific layers*

The general structure of the OSI architecture described in division 5 provides architectural concepts from which the Reference Model of Open Systems Interconnection has been derived, making specific choices for the layers and their contents.

The Reference Model contains seven layers:

a) the Application Layer (layer 7);
b) the Presentation Layer (layer 6);
c) the Session Layer (layer 5);
d) the Transport Layer (layer 4);
e) the Network Layer (layer 3);
f) the Data Link Layer (layer 2); and
g) the Physical Layer (layer 1).

These layers are illustrated in Figure 12/X.200. The highest layer is the Application Layer and it consists of the application-entities that cooperate in the OSI environment. The lower layers provide the services through which the application-entities cooperate.

Layers 1-6, together with the physical media for OSI provide a step-by-step enhancement of communication services. The boundary between two layers identifies a stage in this enhancement of services at which an OSI service Recommendation is defined, while the functioning of the layers is governed by OSI protocol Recommendations.

Not all open systems provide the initial source or final destination of data. When the physical media for OSI do not link all open systems directly, some open systems act only as relay open systems, passing data to other open systems. The functions and protocols which support the forwarding of data are then provided in the lower layers. This is illustrated in Figure 13/X.200.

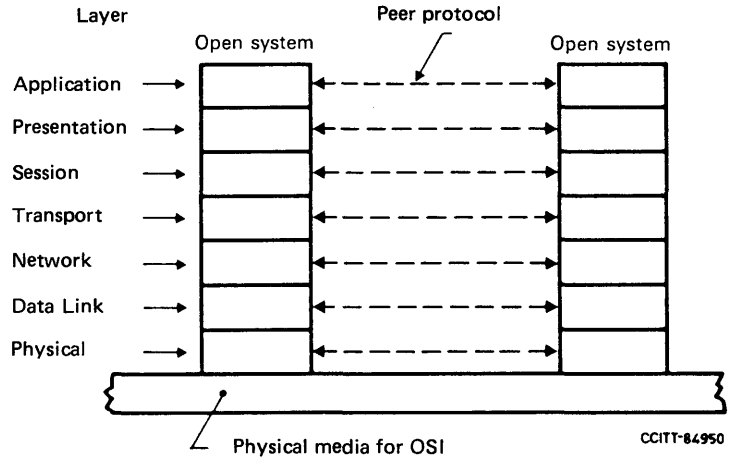


FIGURE 12/X.200

Seven layer reference model and peer protocols

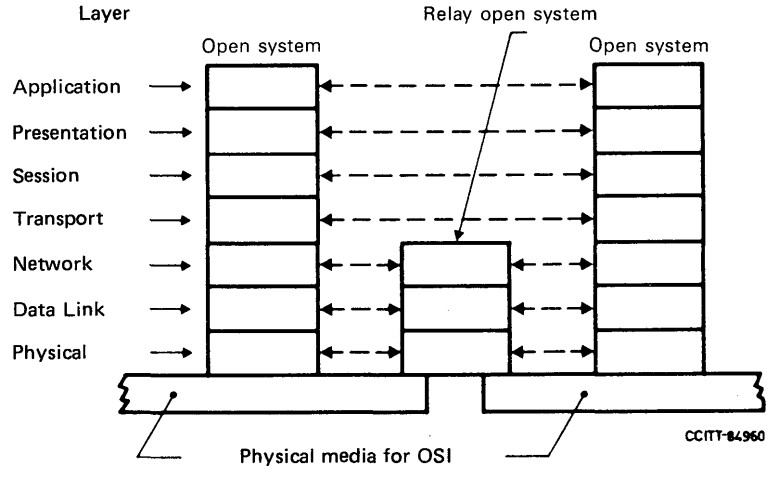


FIGURE 13/X.200

Communication involving relay open systems

6.2 *The principles used to determine the seven layers in the Reference Model*

The following principles have been used to determine the seven layers in the Reference Model and are felt to be useful for guiding further decisions in the development of OSI Recommendations.

Note – It may be difficult to prove that any particular layering selected is the best possible solution. However, there are general principles which can be applied to the question of where a boundary should be placed and how many boundaries should be placed.

P1: do not create so many layers as to make the system engineering task of describing and integrating the layers more difficult than necessary;

P2: create a boundary at a point where the description of services can be small and the number of interactions across the boundary are minimized;

P3: create separate layers to handle functions that are manifestly different in the process performed or the involved technology;

P4: collect similar functions into the same layer;

P5: select boundaries at a point which past experience has demonstrated to be successful;

P6: create a layer of easily localized functions so that the layer could be totally redesigned and its protocols changed in a major way to take advantage of new advances in architectural, hardware or software technology without changing the services expected from and provided to the adjacent layers;

P7: create a boundary where it may be useful at some point in time to have the corresponding interface standardized;

Note 1 – Advantages and drawbacks of standardizing internal interfaces within open systems are not considered in this Recommendation. In particular, mention of, or reference to principle P7, should not be taken to imply usefulness of standards for such internal interfaces.

Note 2 – It is important to note that OSI per se does not require interfaces within open systems to be standardized. Moreover, whenever standards for such interfaces are defined, adherence to such internal interface standards can in no way be considered as a condition of openness.

P8: create a layer where there is a need for a different level of abstraction in the handling of data, e.g., morphology, syntax, semantics;

P9: allow changes of functions or protocols to be made within a layer without affecting other layers; and

P10: create for each layer, boundaries with its upper and lower layer only.

Similar principles have been applied to sublayering:

P11: create further subgrouping and organization of functions to form sublayers within a layer in cases where distinct communications services need it;

P12: create, where needed, two or more sublayers with a common, and therefore minimal functionality to allow interface operation with adjacent layers; and

P13: allow by-passing of sublayers.

6.3 *Layer descriptions*

For each of the seven layers identified above, division 7 provides:

a) an outline of the purpose of the layer;

b) a description of the services offered by the layer to the layer above; and

c) a description of the functions provided in the layer and the use made of the services provided by the layer below.

The descriptions, by themselves, do not provide a complete definition of the services and protocols for each layer. These are the subject of separate Recommendations.

7 Detailed description of the resulting OSI architecture

7.1 *Application layer*

7.1.1 *Definitions*

7.1.1.1 **application-entity**

The aspects of an application-process pertinent to OSI.

7.1.1.2 **application-service-element**

A part of an application-entity which provides an OSI environment capability, using underlying services where appropriate.

7.1.1.3 **user-element**

The representation of that part of the application-process which uses those application-service-elements needed to accomplish the communications objectives of that application-process.

7.1.2 *Purpose*

As the highest layer in the Reference Model of Open Systems Interconnection, the application layer provides a means for the application-processes to access the OSI environment. Hence the application layer does not interface with a higher layer. The application layer is the sole means for the application-processes to access the OSI environment.

The purpose of the application layer is to serve as the window between correspondent application-processes which are using the OSI to exchange meaningful information.

Each application-process is represented to its peer by the application-entity.

All specifiable application-process parameters of each OSI environment communications instance are made known to the OSI environment (and, thus, to the mechanisms implementing the OSI environment) via the application layer.

7.1.3 *Services provided to applications-processes*

Application-processes exchange information by means of application-entities, application-protocols, and presentation services.

As the only layer in the Reference Model that directly provides services to the application-processes, the application layer necessarily provides all *OSI* services directly usable by application-processes.

The application-entity contains one user-element and a set of application-service-elements. The user-element represents that part of the application-process which uses those application-service-elements needed to accomplish the communications objectives of that application-process. Application-service-elements may call upon each other and/or upon presentation services to perform their function.

The only means by which user-elements in different systems may communicate is through the exchange of application-protocol-data-units. These application-protocol-data-units are generated by application-service-elements.

Note – The application services differ from services provided by other layers in neither being provided to an upper layer nor being associated with a service-access-point.

In addition to information transfer, such services may include, but are not limited to the following:

Note – Some of the services listed below are provided by OSI management.

a) identification of intended communications partners (e.g., by name, by address, by definite description, by generic description);

b) determination of the current availability of the intended communication partners;

c) establishment of the authority to communicate;

d) agreement on privacy mechanisms;

e) authentication of intended communication partners;

f) determination of cost allocation methodology;

g) determination of the adequacy of resources;

h) determination of the acceptable quality of service (e.g., response time, tolerable error rate, cost vis-a-vis the previous considerations);

i) synchronization of cooperating applications;

j) selection of the dialogue discipline including the initiation and release procedures;

k) agreement on the responsibility for error recovery;

l) agreement on procedures for control of data integrity; and

m) identification of constraints on data syntax (character sets, data structure).

7.1.4 *Functions within the application layer*

The application layer contains all functions which imply communication between open-systems and are not already performed by the lower layers. These include functions performed by programs as well as functions performed by human beings.

When a specific instance of an application-process wishes to communicate with an instance of an application-process in some other open-system, it must invoke an instance of an application-entity in the application layer of its own open-system. It then becomes the responsibility of this instance of the application entity to establish an association with an instance of an appropriate application-entity in the destination open-system. This process occurs by invocation of instances of entities in the lower layers. When the association between the two application-entities has been established, the application-processes can communicate.

7.1.4.1 *Grouping of functions in the application layer*

An application-entity can be structured internally into groups of functions. The technique used to express this structure is not constrained by this Recommendation. Use of one grouping of functions may depend on use of some other functions, and the active functions may vary during the lifetime of an association.

The structuring of application-entities into application-service-elements and the user-element provides an organization of functions in application-entities. Furthermore, any given subset of application-service-elements, along with the user-element, constitutes an application-entity type. Each application-entity type, and each instance thereof, are unambiguously identifiable.

An application-process may determine the grouping of functions comprising the application-entity.

Two categories of application-service-elements are recognized: common-application-service-elements and specific-application-service-elements. Common-application-service-elements provide capabilities that are generally useful to a variety of applications. Specific-application-service-elements provide capabilities required to satisfy the particular needs of specific applications (for example, file-transfer, data base access, job transfer, banking, order-entry). Application entities may contain application-service-elements from both categories, as illustrated in Figure 14/X.200.

The partitioning of application-service-elements into these two categories does not imply the existence of two independent protocols.

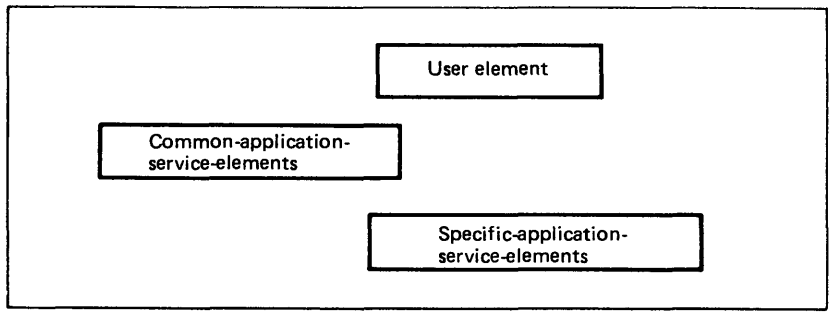


FIGURE 14/X.200

Application-entity

7.1.4.2 *Systems-management and application-management*

Systems-management functions and application-management functions are located in the application layer. For details see sub-division 5.9.

7.1.4.3 *Application layer management*

In addition to systems- and application-management, there are other activities specifically related to application layer management (such as activation and error control). See sub-division 5.9 for the relationship with other management aspects.

7.2 *Presentation layer*

7.2.1 *Definitions*

7.2.1.1 **concrete syntax**

Those aspects of the rules used in the formal specification of data which embody a specific representation of that data.

7.2.1.2 **transfer syntax**

That concrete syntax used in the transfer of data between open systems.

7.2.2 *Purpose*

The presentation layer provides for the representation of information that application-entities either communicate or refer to in their communication.

The presentation layer covers two complementary aspects of this representation of information:

a) the representation of data to be transferred between application-entities; and

b) the representation of the data structure which application-entities refer to in their communication, along with the representations of the set of actions which may be performed on this data structure.

The complementary aspects of the representation of information outlined above refer to the general concept of transfer syntax.

The presentation layer is concerned only with the syntax, (i.e., the representation of the data) and not with its semantics, (i.e., their meaning to the application layer), which is known only by the application-entities.

The presentation layer provides for a common representation to be used between application-entities. This relieves application-entities of any concern with the problem of "common" representation of information, i.e., it provides them with syntax independence. This syntax independence can be described in two ways:

a) the presentation layer provides common syntactical elements which are used by application-entities; and

b) the application-entities can use any syntax and the presentation layer provides the transformation between these syntaxes and the common syntax needed for communication between application-entities. This transformation is performed inside the open systems. It is not seen by other open systems and therefore has no impact on the standardization of presentation-protocols.

In this Recommendation the approach outlined in b) is used.

7.2.3 *Services provided to the application layer*

The presentation layer provides session-services (see § 7.3) and the following facilities:

a) transformation of syntax; and

b) selection of syntax.

Tranformation of syntax is concerned with code and character set conversions, with the modification of the layout of the data and the adaptation of actions on the data structures. Selection of syntax provides the means of initially selecting a syntax and subsequently modifying the selection.

Session-services are provided to application entities in the form of presentation-services.

7.2.4 *Functions within the presentation layer*

The presentation layer peforms the following functions to help accomplish the presentation-services:

a) session establishment request;

b) data transfer;

c) negotiation and renegotiation of syntax;

d) transformation of syntax including data transformation, formatting and special purpose transformations (e.g., compression); and

e) session termination request.

7.2.4.1 *Transformation of syntax*

The fact that there is or is not actual transformation of syntax has no impact on the presentation protocol.

There are three syntactic versions of the data: the syntax used by the originating application-entity, the syntax used by the receiving application-entity and the syntax used between presentation-entities (the transfer syntax). It is clearly possible that any two or all three of these syntaxes may be identical. The presentation layer contains the functions necessary to transform between the transfer syntax and each of the other syntaxes as required.

There is not a single predetermined transfer syntax for all OSI. The transfer syntax to be used on a presentation-connection is negotiated between the correspondent presentation-entities. Thus, a presentation-entity must know the syntax of its application-entity and the agreed transfer syntax. Only the transfer syntax needs to be referred to in the presentation layer protocols.

To meet the service requirements specified by the application-entities during the intitiation phase, the presentation layer may utilize any transfer syntax available to it. To accomplish other service objectives (e.g., data volume reduction to reduce data transfer cost), syntax transformation may be performed either as a specific syntax-matching service provided to the application-entities, or as a function internal to the presentation layer.

7.2.4.2 *Negotiation of syntax*

Negotiation of syntax is carried out by communication between presentation-entities on behalf of the application-entities to determine the form that data will have while in the OSI environment. The negotiations will determine what transformations are needed (if any) and where they will be performed. Negotiations may be limited to the initiation phase or they may occur any time during a session.

In OSI, the syntaxes used by application-entities that wish to communicate may be very similar or quite dissimilar. When they are similar, the transformation functions may not be needed at all; however, when they are dissimilar, the presentation layer services provide the means to converse and decide where needed transformations will take place.

7.2.4.3 *Addressing and multiplexing*

There is a one-to-one correspondence between presentation-address and session-address. There is no multiplexing or splitting in the presentation layer.

7.2.4.4 *Presentation layer management*

The presentation layer protocols deal with some management activities of the layer (such as activation and error control). See sub-division 5.9 for relationship with other management aspects.

7.3 *Session layer*

7.3.1 *Definitions*

7.3.1.1 **quarantine service**

A facility of the session-service by which an integral number of session-service-data-units sent on a session-connection are not made available to the receiving presentation-entity until explicitly released by the sending presentation-entity.

7.3.1.2 **interaction management**

A facility of the session-service which allows correspondent presentation-entities to control explicitly whose turn it is to exercise certain control functions.

7.3.1.3 **two-way-simultaneous interaction**

A mode of interaction where both presentation-entities may concurrently send and receive.

7.3.1.4 **two-way-alternate interaction**

A mode of interaction where the presentation-entity with the turn may send and its correspondent is permitted only to receive.

7.3.1.5 **one-way interaction**

A form of operation of two-way-alternate interaction in which the turn can never be exchanged.

7.3.1.6 **session-connection synchronization**

A facility of the session-service which allows presentation-entities to define and identify synchronization points and to reset a session-connection to a predefined state and to agree on a resynchronization point.

7.3.2 *Purpose*

The purpose of the session layer is to provide the means necessary for cooperating presentation-entities to organize and synchronize their dialogue and to manage their data exchange. To do this, the session layer provides services to establish a session-connection between two presentation-entities, and to support orderly data exchange interactions.

To implement the transfer of data between the presentation-entities, the session-connection is mapped onto and uses a transport-connection (see § 7.3.4.1).

A session-connection is created when requested by a presentation-entity at a session-service-access-point. During the lifetime of the session-connection, session services are used by the presentation-entities to regulate their dialogue, and to ensure an orderly message exchange on the session-connection. The session-connection exists until it is released by either the presentation-entities or the session-entities. While the session-connection exists, session services maintain the state of the dialogue even over data loss by the transport layer.

A presentation-entity can access another presentation-entity only by initiating or accepting a session-connection. A presentation-entity may be associated with several session-connections simultaneously. Both concurrent and consecutive session-connections are possible between two presentation-entities.

The initiating presentation-entity designates the destination presentation-entity by a session-address. In many systems, a transport-address may be used as the session-address, i.e., there is a one-to-one correspondence between the session-address and the transport-address. In general, however, there is a many-to-one correspondence between session-addresses and transport-address. This does not imply multiplexing of session-connections onto transport-connections, but does imply that at session-connection establishment time, more than one presentation-entity is a potential target of a session-connection establishment request arriving on a given transport-connection.

7.3.3 *Services provided to the presentation layer*

The following services provided by the session layer are described below:

a) session-connection establishment;
b) session-connection release;
c) normal data exchange;
d) quarantine service;
e) expedited data exchange;
f) interaction management;
g) session-connection synchronization; and
h) exception reporting.

7.3.3.1 *Session-connection establishment*

The session-connection establishment service enables two presentation-entities to establish a session-connection between themselves. The presentation-entities are identified by session-addresses used to request the establishment of the session-connection.

The session-connection establishment service allows the presentation-entities cooperatively to determine the unique values of session-connection parameters at the time the session-connection is established.

Note – The provision for change of session parameters after session-connection establishment is a candidate for further extension.

Simultaneous session-connection establishment requests typically result in a corresponding number of session-connections, but a session-entity can always reject an incoming request.

The session-connection establishment service provides to the presentation-entities a session-service-connection identifier which uniquely specifies the session-connection within the environment of the correspondent presentation-entities, with a lifetime which may be greater than the lifetime of the session-connection. This identifier may be used by the presentation- entities, to refer to the session-connection during the lifetime of the session-connection, and may also be used by management-entities for administrative purposes such as accounting.

7.3.3.2 *Session-connection release*

The session-connection release service allows presentation-entities to release a session-connection in an orderly way without loss of data. It also allows either presentation-entity to request at any time that a session-connection be aborted; in this case, data may be lost.

The release of a session-connection may also be initiated by one of the session-entities supporting it.

7.3.3.3 *Normal data exchange*

The normal data exchange service allows a sending presentation-entity to transfer a session-service-data-unit to a receiving presentation-entity. This service allows the receiving presentation-entity to ensure that it is not overloaded with data.

7.3.3.4 *Quarantine service*

The quarantine service allows the sending presentation-entity to request that an integral number of session-service-data-units (one or more) sent on a session-connection should not be made available to the receiving presentation-entity until explicitly released by the sending presentation-entity. The sending presentation-entity may request that all data currently quarantined be discarded. The receiving presentation-entity receives no information that data being received has been quarantined or that some data was discarded.

7.3.3.5 *Expedited data exchange*

The expedited data exchange service provides expedited handling for the transfer of expedited session-service-data-units. A specific size restriction is placed on expedited session-service-data-units. This service may be used by either presentation-entity at any time that a session-connection exists.

7.3.3.6 *Interaction management*

The interaction management service allows the presentation-entities to control explicitly whose turn it is to exercise certain control functions.

The service provides for voluntary exchange of the turn where the presentation-entity which has the turn relinquishes it voluntarily. This service also provides for forced exchange of the turn where, upon request from the presentation-entity which does not have the turn, the session-service may force the presentation-entity with the turn to relinquish it. In the case of forced exchange of the turn, data may be lost.

The following types of session-service-data-unit exchange interaction are defined:

a) two-way-simultaneous (TWS);

b) two-way-alternate (TWA); and

c) one-way interaction.

7.3.3.7 *Session-connection synchronization*

The session-connection synchronization service allows presentation-entities to:

a) define and identify synchronization points; and

b) reset the session-connection to a defined state and agree on a resynchronization point.

The session layer is not responsible for any associated checkpointing or commitment action associated with synchronization.

7.3.3.8 *Exception reporting*

The exception reporting service permits the presentation-entities to be notified of exceptional situations not covered by other services, such as unrecoverable session malfunctions.

Note – The following services are candidates for future extensions:

a) session-service-data-unit sequence numbering;

b) brackets;

c) stop-go; and

d) security.

7.3.4 *Functions within the session layer*

The functions within the session layer are those which are performed by session-entities in order to provide the session services.

Most of the functions required are readily implied by the services provided. Additional description is given below for the following functions:

a) session-connection to transport-connection mapping;

b) session-connection flow control;

c) expedited data transfer;

d) session-connection recovery;

e) session-connection release; and

f) session layer management.

7.3.4.1 *Session-connection to transport-connection mapping*

There is a one-to-one mapping between a session-connection and a transport-connection at any given instant. However, the lifetime of a transport-connection and that of a related session-connection can be distinguished so that the following cases are defined:

a) a transport-connection supports several consecutive session-connections (see Figure 15/X.200); and

b) several consecutive transport-connections support a session-connection (see Figure 16/X.200).

Note 1 – It is also possible to consider cases in which one transport-connection is used to support several session-connections (i.e., n-to-1 mapping). In this case peer flow control would be required in the session layer. This case is for future development if needed.

Note 2 – To implement the mapping of a session-connection onto a transport-connection, the session layer maps session-service-data-units into session-protocol-data-units, and session-protocol-data-units into transport-service-data-units. These mappings may require the session-entities to perform functions such as segmenting. These functions are visible only in the session-protocols, therefore they are transparent to the presentation and transport layers.

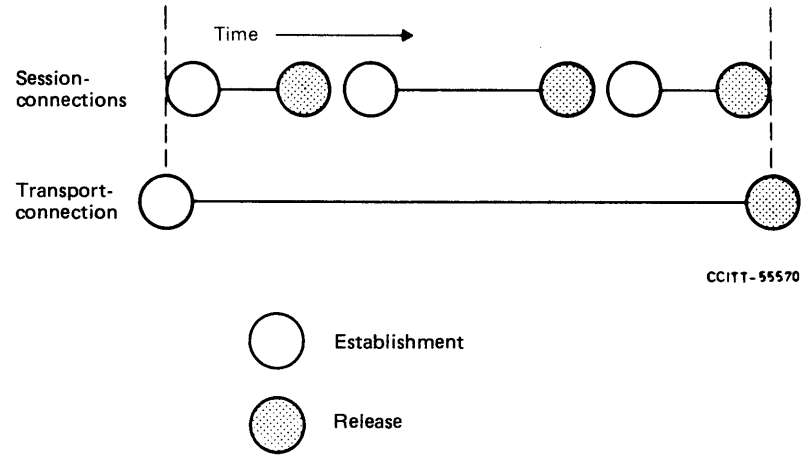


FIGURE 15/X.200

Several consecutive session-connections

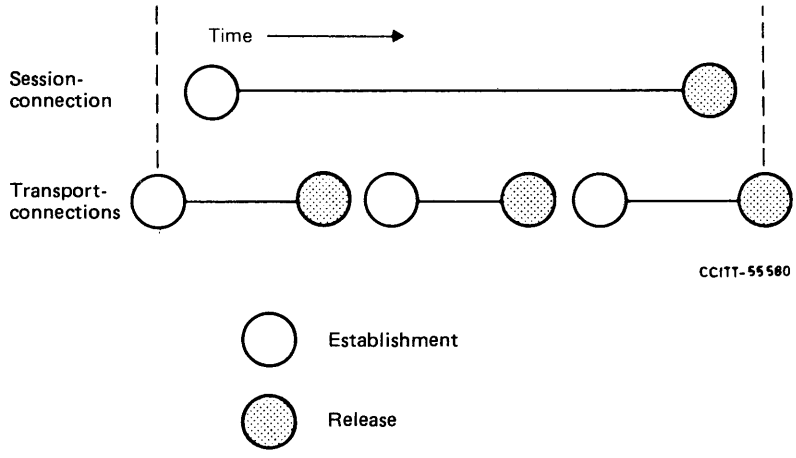


FIGURE 16/X.200

Several consecutive transport-connections

7.3.4.2 *Session-connection flow control*

There is no peer flow control in the session layer. To prevent the receiving presentation-entity from being overloaded with data, the receiving session-entity applies back pressure across the transport-connection using the transport flow control.

7.3.4.3 *Expedited data transfer*

The transfer of expedited session-service-data-units is generally accomplished by use of the expedited transport service.

7.3.4.4 *Session-connection recovery*

In the event of reported failure of an underlying transport-connection, the session layer may contain the functions necessary to re-establish a transport connection to support the session-connection, which continues to exist. The session-entities involved notify the presentation-entities via the exception reporting service that service is interrupted and restore the service only as directed by the presentation-entities. This permits the presentation-entities to resynchronize and continue from an agreed state.

7.3.4.5 *Session-connection release*

The session layer contains the functions necessary to release the session-connection in an orderly way, without loss of data, upon request by the presentation-entities. The session layer also contains the necessary functions to abort the session-connection with the possible loss of data.

7.3.4.6 *Session layer management*

The session layer protocols deal with some management activities of the layer (such as activation and error control). See sub-division 5.9 for the relationship with other management aspects.

7.4 *The transport layer*

7.4.1 *Definitions*

No transport layer specific terms are identified.

7.4.2 *Purpose*

The transport-service provides transparent transfer of data between session-entities and relieves them from any concern with the detailed way in which reliable and cost effective transfer of data is achieved.

The transport layer optimizes the use of the available network-service to provide the performance required by each session-entity at minimum cost. This optimization is achieved within the constraints imposed by the overall demands of all concurrent session-entities and the overall quality and capacity of the network-service available to the transport layer.

All protocols defined in the transport layer have end-to-end significance, where the ends are defined as correspondent transport-entities. Therefore the transport layer is OSI end open system oriented and transport-protocols operate only between OSI end open systems.

The transport layer is relieved of any concern with routing, and relaying since the network-service provides network-connections from any transport-entity to any other, including the case of tandem subnetworks (see § 7.5.1).

The transport functions invoked in the transport layer to provide a requested service quality depend on the quality of the network-service. The quality of the network-service depends on the way the network-service is achieved (see § 7.5.3).

7.4.3 *Services provided to the session layer*

The transport layer uniquely identifies each session-entity by its transport-address. The transport-service provides the means to establish, maintain and release transport-connections. Transport-connections provide duplex transmission between a pair of transport-addresses.

More than one transport-connection can be established between the same pair of transport-addresses. A session-entity uses transport-connection-endpoint-identifiers provided by the transport layer to distinguish between transport-connection-endpoints.

The operation of one transport-connection is independent of the operation of all others except for the limitations imposed by the finite resources available to the transport layer.

The quality of service provided on a transport-connection depends on the service class requested by the session-entities when establishing the transport-connection. The selected quality of service is maintained throughout the lifetime of the transport-connection. The session-entity is notified of any failure to maintain the selected quality of service on a given transport-connection.

The following services provided by the transport layer are described below:

a) transport-connection establishment;

b) data transfer; and

c) transport-connection release.

7.4.3.1 *Transport-connection establishment*

Transport-connections are established between session-entities identified by transport-addresses. The quality of service of the transport-connection is negotiated between the session-entities and the transport-service.

At the time of establishment of a transport-connection the class of transport service to be provided can be selected from a defined set of available classes of service.

These service classes are characterized by combinations of selected values of parameters such as throughput, transit delay, and connection set-up delay and by guaranteed values of parameters such as residual error rate and service availability.

These classes of service represent globally predefined combinations of parameters controlling quality of service. These classes of service are intended to cover the transport-service requirements of the various types of traffic generated by the session-entities.

7.4.3.2 *Data transfer*

This service provides data transfer in accordance with the agreed quality of service. When the quality of service cannot be maintained and all possible recovery attempts have failed, the transport-connection is terminated and the session-entities are notified.

a) The transport-service-data-unit transfer-service provides the means by which transport-service-data-units of arbitrary length are delimited and transparently transferred in sequence from one sending transport-service-access-point to the receiving transport-service-access-point over a transport-connection. This service is subject to flow control.

b) The expedited transport-service-data-unit transfer service provides an additional means of information exchange on a transport-connection. The expedited transport-data-units are subject to their own set of transport-service and flow control characteristics. The maximum size of expedited transport-service data-units is limited.

7.4.3.3 *Transport-connection release*

This service provides the means by which either session-entity can release a transport-connection and have the correspondent session-entity informed of the release.

7.4.4 *Functions within the transport layer*

The transport layer functions may include:

a) mapping transport-address onto network-address;

b) multiplexing (end-to-end) transport-connections onto network-connections;

c) establishment and release of transport-connections;

d) end-to-end sequence control on individual connections;

e) end-to-end error detection and any necessary monitoring of the quality of service;

f) end-to-end error recovery;

g) end-to-end segmenting, blocking and concatenation;

h) end-to-end flow control on individual connections;

i) supervisory functions; and

j) expedited transport-service-data-unit transfer.

7.4.4.1 *Addressing*

When a session-entity requests the transport layer to establish a transport-connection with another session-entity identified by its transport-address, the transport layer determines the network-address identifying the transport-entity which serves the correspondent session-entity.

Because transport-entities support services on an end-to-end basis no intermediate transport-entity is involved as a relay between the end transport-entities. Therefore the transport layer maps transport-addresses to the network-addresses which identify the end transport-entities (see Figure 17/X.200).

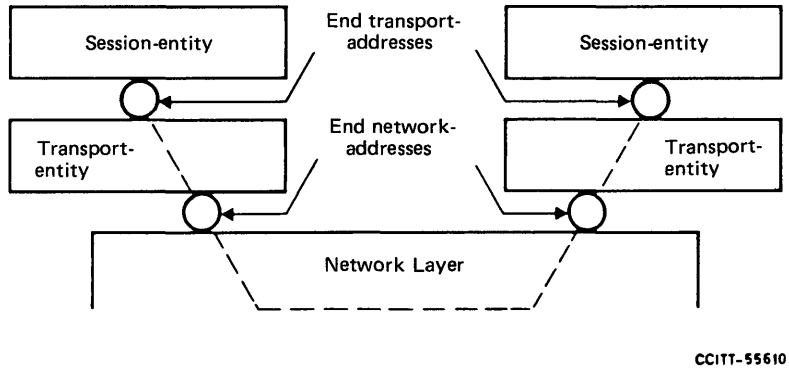


FIGURE 17/X.200

Association of transport-addresses and network-addresses

One transport-entity may serve more than one session-entity. Several transport-addresses may be associated with one network-address within the scope of the same transport-entity. Corresponding mapping functions are perfomed within the transport-entities to provide these facilities (see Figure 18/X.200).

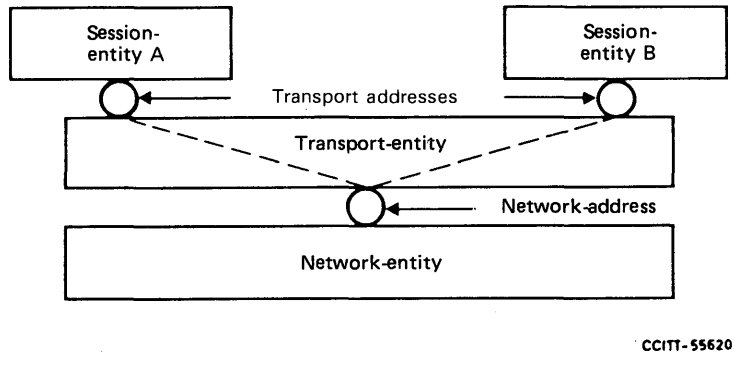


FIGURE 18/X.200

Association of one network-address with several transport-addresses

7.4.4.2 *Connection multiplexing and splitting*

In order to optimize the use of network-connections, the mapping of transport-connections onto network-connections need not be on a one-to-one basis. Both splitting and multiplexing may be performed, namely for optimizing cost of usage of the network-service.

7.4.4.3 *Phases of operation*

The phases of operation within the transport layer are:

a) establishment phase;

b) data transfer phase; and

c) release phase.

The transfer from one phase of operation to another will be specified in detail within the protocol for the transport layer.

7.4.4.4 *Establishment phase*

During the establishment phase, the transport layer establishes a transport-connection between two session-entities. The functions of the transport layer during this phase match the requested class of services with the services provided by the network layer. The following functions may be performed during this phase:

a) obtain a network-connection which best matches the requirements of the session-entity taking into account cost and quality of service;

b) decide whether multiplexing or splitting is needed to optimize the use of network connections;

c) establish the optimum transport-protocol-data-unit size;

d) select the functions that will be operational upon entering the data transfer phase;

e) map transport-addresses onto network-addresses;

f) provide identification of different transport-connections between the same pair of transport-service-access-points (connection identification function); and

g) transfer of data.

7.4.4.5 *Data transfer phase*

The purpose of the data transfer phase is to transfer transport-service-data-units between the two session-entities connected by the transport-connection. This is achieved by the transmission of transport-protocol-data-units and by the following functions, each of which is used or not used according to the class of service selected in the establishment phase:

a) sequencing;

b) blocking;

c) concatenation;

d) segmenting;

e) multiplexing or splitting;

f) flow control;

g) error detection;

h) error recovery;

i) expedited data transfer;

j) transport-service-data-unit delimiting; and

k) transport-connection identification.

7.4.4.6 *Release phase*

The purpose of the release phase is to release the transport-connection. It may include the following functions:

a) notification of reason for release;

b) identification of the transport-connection released; and

c) transfer of data.

7.4.4.7 *Transport layer management*

The transport layer protocols deal with some management activities of the layer (such as activation and error control). See sub-division 5.9 for the relationship with other management aspects.

7.5 *Network layer*

7.5.1 *Definitions*

7.5.1.1 **subnetwork**

A set of one or more intermediate open systems which provide relaying and through which end systems may establish network-connections.

Note – A subnetwork is a representation within the OSI Reference Model of a real network such as a carrier network, a private network or a local area network.

7.5.1.2 **subnetwork-connection**

A communication path through a subnetwork which is used by entities in the network layer in providing a network-connection.

7.5.2 *Purpose*

The network layer provides the means to establish, maintain and terminate network-connections between open systems containing communicating application-entities and the functional and procedural means to exchange network-service-data-units between transport-entities over network connections.

It provides to the transport-entities independence from routing and relay considerations associated with the establishment and operation of a given network-connection. This includes the case where several subnetworks are used in tandem (see § 7.5.4.2) or in parallel. It makes invisible to transport-entities how underlying resources such as data-link-connections are used to provide network-connections.

Any relay functions and hop-by-hop service enhancement protocols used to support the network-service between the OSI end open systems are operating below the transport layer, i.e., within the network layer or below.

7.5.3 *Services provided to the transport layer*

The basic service of the network layer is to provide the transparent transfer of data between transport-entities. This service allows the structure and detailed content of submitted data to be determined exclusively by layers above the network layer.

All services are provided to the transport layer at a known cost.

The network layer contains functions necessary to provide the transport layer with a firm network/transport layer boundary which is independent of the underlying communications media in all things other than quality of service. Thus the network layer contains functions necessary to mask the differences in the characteristics of different transmission and subnetwork technologies into a consistent network service.

The service provided at each end of a network-connection is the same even when a network-connection spans several subnetworks, each offering dissimilar services (see § 7.5.4.2).

Note – It is important to distinguish the specialized use of the term "service" within the OSI Reference Model from its common use by suppliers of private networks and carriers.

The quality of service is negotiated between the transport-entities and the network-service at the time of establishment of a network-connection. While this quality of service may vary from one network-connection to another it will be agreed for a given network-connection and be the same at both network-connection-endpoints.

The following services or elements of services provided by the network layer are described below:

a) network-addresses;

b) network-connections;

c) network-connection-endpoint-identifiers;

d) network-service-data-unit transfer;

e) quality of service parameters;

f) error notification;

g) sequencing;

h) flow control;

i) expedited network-service-data-unit transfer;

j) reset;

k) release; and

l) receipt of confirmation.

Some of the services described below are optional. This means that:

a) the user has to request the service; and

b) the network-service provider may honour the request or indicate that the service is not available.

7.5.3.1 *Network-addresses*

Transport-entities are known to the network layer by means of network-addresses. Network-addresses are provided by the network layer and can be used by transport-entities to identify uniquely other transport-entities, i.e., network addresses are necessary for transport-entities to communicate using the network-service. The network layer uniquely identifies each of the end open systems (represented by transport-entities) by their network-addresses. This may be independent of the addressing needed by the underlying layers.

7.5.3.2 *Network-connections*

A network-connection provides the means of transferring data between transport-entities identified by network-addresses. The network layer provides the means to establish, maintain and release network-connections.

A network-connection is point-to-point.

More than one network-connection may exist between the same pair of network-addresses.

7.5.3.3 *Network-connection-endpoint-identifiers*

The network layer provides to the transport-entity a network-connection-endpoint-identifier which identifies the network-connection-end-point uniquely with the associated network-address.

7.5.3.4 *Network-service-data-unit transfer*

On a network-connection, the network layer provides for the transmission of network-service-data-units. These units have a distinct beginning and end and integrity of the unit's content is maintained by the network layer.

No limit is imposed on the maximum size of network-service-data-units.

The network-service-data-units are transferred transparently between transport-entities.

7.5.3.5 *Quality of service parameters*

The network layer establishes and maintains a selected quality of service for the duration of the network-connection.

The quality of service parameters include residual error rate, service availability, reliability, throughput, transit delay (including variations), and delay for network-connection establishment.

7.5.3.6 *Error notification*

Unrecoverable errors detected by the network layer are reported to the transport-entities.

Error notification may or may not lead to the release of the network-connection, according to the specification of a particular network-service.

7.5.3.7 *Sequencing*

The network layer may provide sequenced delivery of network-service-data-units over a given network-connection when requested by the transport-entities.

7.5.3.8 *Flow control*

A transport-entity which is receiving at one end of a network-connection can cause the network-service to stop transferring network-service-data-units across the service-access-point. This flow control condition may or may not be propagated to the other end of the network-connection and thus be reflected to the transmitting transport-entity, according to the specification of a particular network-service.

7.5.3.9 *Expedited network-service-data-unit transfer (optional)*

The expedited network-service-data-unit transfer is optional and provides an additional means of information exchange on a network-connection. The transfer of expedited network-service-data-units is subject to a different set of network-service characteristics and to separate flow control.

The maximum size of expedited network-service-data-units is limited.

7.5.3.10 *Reset (optional)*

The reset service is optional and when invoked causes the network layer to discard all network-service-data-units in transit on the network-connection and to notify the transport-entity at the other end of the network-connection that a reset has occurred.

7.5.3.11 *Release*

A transport-entity may request release of a network-connection. The network-service does not guarantee the delivery of data preceding the release request and still in transit. The network-connection is released regardless of the action taken by the correspondent transport-entity.

7.5.3.12 *Receipt of confirmation (optional)*

A transport-entity may confirm receipt of data over a network-connection. The use of receipt confirmation service is agreed by the two users of the network-connection during connection establishment.

This service is an optional service that may not always be available.

Note – This service is included in the network service only to support existing features of CCITT Recommendation X.25.

7.5.4 *Functions within the network layer*

Network layer functions provide for the wide variety of configurations supporting network-connections ranging from network-connections supported by point-to-point configurations, to network-connections supported by complex combinations of subnetworks with different characteristics.

Note – In order to cope with this wide variety of cases, network functions should be structured into sublayers. The subdivision of the network layer into sublayers need only be done when this is useful. In particular, sublayering need not be used when the access protocol of the subnetwork supports the complete functionality of the OSI network service.

The following are functions performed by the network layer:

a) routing and relaying;
b) network-connections;
c) network-connection multiplexing;
d) segmenting and blocking;
e) error detection;
f) error recovery;
g) sequencing;
h) flow control;
i) expedited data transfer;
j) reset;
k) service selection; and
l) network layer management.

7.5.4.1 *Routing and relaying*

Network-connections are provided by network-entities in end open systems but may involve intermediate open systems which provide relaying. These intermediate open systems may interconnect subnetwork-connections, data-link-connections, and data-circuits (see § 7.7). Routing functions determine an appropriate route between network-addresses. In order to set up the resulting communication, it may be necessary for the network layer to use the services of the data link layer to control the interconnection of data-circuits (see §§ 7.6.4.10 and 7.7.3.1).

The control of interconnection of data-circuits (which are in the physical layer) from the network layer requires interaction between a network-entity and a physical entity in the same open system. Since the Reference Model permits direct interaction only between adjacent layers, the network-entity cannot interact directly with the physical-entity. This interaction is thus described as through the data link layer which intervenes "transparently" to convey the interaction between the network layer and the physical layer.

This representation is an abstract representation of something happening inside an open system and which does not model the functioning of real open systems and as such has no impact on the standardization of OSI protocols.

Note – When network layer functions are performed by combinations of several individual subnetworks, the specification of routing and relaying functions could be facilitated by using sublayers, isolating individual subnetworks routing and relaying functions from internetwork routing and relaying functions. However, when subnetworks have access protocols supporting the complete functionality of the OSI network service, there need be no sublayering in the network layer.

7.5.4.2 *Network-connections*

This function provides network-connections between transport-entities, making use of data-link-connections provided by the data link layer.

A network-connection may also be provided as subnetwork-connections in tandem, i.e., using several individual subnetworks in series. The interconnected individual subnetworks may have the same or different service capabilities. Each end of a subnetwork-connection may operate with a different subnetwork protocol.

The interconnection of a pair of subnetworks of differing qualities may be achieved in two ways. To illustrate these, consider a pair of subnetworks, one of high quality and the other of low quality:

a) The two subnetworks are interconnected as they stand. The quality of the resulting network-connection is not higher than that of the lower quality subnetwork (see Figure 19/X.200).

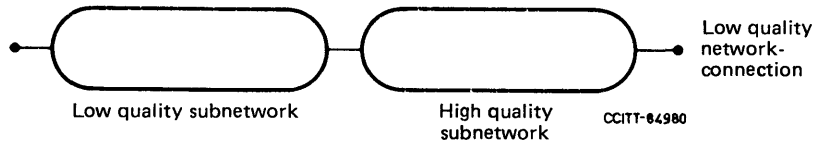


FIGURE 19/X.200

Interconnection of a low quality subnetwork and a high quality subnetwork

b) The lower quality subnetwork is enhanced equal to the higher quality subnetwork and the subnetworks are then interconnected. The quality of the resulting network-connection is aproximately that of the higher quality subnetwork (see Figure 20/X.200).

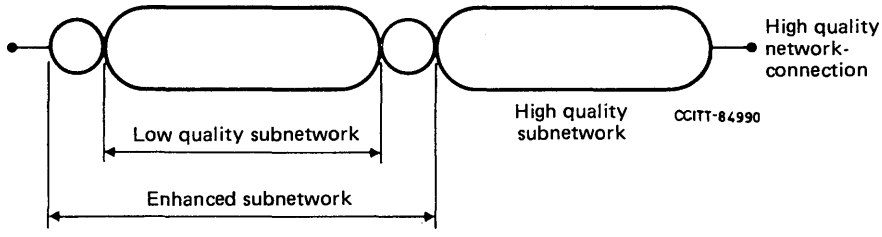


FIGURE 20/X.200

Interconnection of an enhanced low quality subnetwork and a high quality subnetwork

The choice between these two alternatives depends on the degree of difference in quality, the cost of enhancement, and other economic factors.

7.5.4.3 *Network-connection multiplexing*

This function may be used to multiplex network-connections onto data-link-connections in order to optimize their use.

In the case of subnetwork-connections in tandem, multiplexing onto individual subnetwork-connections may also be performed in order to optimize their use.

7.5.4.4 *Segmenting and blocking*

The network layer may segment and/or block network-service-data-units for the purpose of facilitating the transfer. However, the network-service-data-unit delimiters are preserved over the network-connection.

7.5.4.5 *Error detection*

Error detection functions are used to check that the quality of service provided over a network-connection is maintained. Error detection in the network layer uses error notification from the data link layer. Additional error detection capabilities may be necessary to provide the required quality of service.

7.5.4.6 *Error recovery*

This function provides for recovering from detected errors. This function may vary depending on the quality of the network service provided.

7.5.4.7 *Sequencing*

This function provides for the sequenced delivery of network-service-data-units over a given network-connection when requested by transport-entities.

7.5.4.8 *Flow control*

If flow control service is required (see § 7.5.3.8), this function may need to be performed.

7.5.4.9 *Expedited data transfer*

This function provides for the expedited data transfer service.

7.5.4.10 *Reset*

This function provides for the reset service.

7.5.4.11 *Service selection*

This function allows service selection to be carried out to ensure that the service provided at each end of a network-connection is the same when a network-connection spans several subnetworks of dissimilar quality.

7.5.4.12 *Network layer management*

The network layer protocols deal with some management activities of the layer (such as activation and error control). See sub-division 5.9 for the relationship with other management aspects.

7.6 *Data link layer*

7.6.1 *Definitions*

No data link layer specific terms are identified.

7.6.2 *Purpose*

The data link layer provides functional and procedural means to establish, maintain and release data-link-connections among network-entities and to transfer data-link-service-data-units. A data-link-connection is built upon one or several physical-connections.

The data link layer detects and possibly corrects errors which may occur in the physical layer.

In addition, the data link layer enables the network layer to control the interconnection of data-circuits within the physical layer.

7.6.3 *Services provided to the network layer*

The following services or elements of services provided by the data link layer are described below:

a) data-link-connection;

b) data-link-service-data-units;

c) data-link-connection-endpoint-identifiers;

d) sequencing;

e) error notification;

f) flow control; and

g) quality of service parameters.

7.6.3.1 *Data-link-connection*

The data link layer provides one or more data-link-connections between two network-entities. A data-link-connection is always established and released dynamically.

7.6.3.2 *Data-link-service-data-units*

The data link layer allows exchange of data-link-service-data-units over a data-link-connection.

The size of the data-link-service-data-units may be limited by the relationship between the physical-connection error rate and the data link layer error detection capability.

7.6.3.3 *Data-link-connection-endpoint-identifiers*

If needed, the data link layer provides data-link-connection-endpoint-identifiers that can be used by a network-entity to identify a correspondent network-entity.

7.6.3.4 *Sequencing*

When required, the sequence integrity of data-link-service-data-units is maintained.

7.6.3.5 *Error notification*

Notification is provided to the network-entity when any unrecoverable error is detected by the data link layer.

7.6.3.6 *Flow control*

Each network-entity can dynamically control (up to the agreed maximum) the rate at which it receives data-link-service-data-units from a data-link-connection. This control may be reflected in the rate at which the data link layer accepts data-link-service-data-units at the correspondent data-link-connection-endpoint.

7.6.3.7 *Quality of service parameters*

Quality of service parameters may be optionally selectable. The data link layer establishes and maintains a selected quality of service for the duration of the data-link-connection. The quality of service parameters include mean time between detected but unrecoverable errors, residual error rate (where errors may arise from alteration, loss, duplication, disordering, misdelivery of data-link-service-data-unit, and other causes), service availability, transit delay and throughput.

7.6.4 *Functions within the data link layer*

The following functions performed by the data link layer are described below:

a) data-link-connection establishment and release;
b) data-link-service-data-unit mapping;
c) data-link-connection splitting;
d) delimiting and synchronization;
e) sequence control;
f) error detection;
g) error recovery;
h) flow control;
i) identification and parameter exchange;
j) control of data-circuit interconnection; and
k) data link layer management.

7.6.4.1 *Data-link-connection establishment and release*

This function establishes and releases data-link-connections on activated physical-connections. When a physical-connection has multiple endpoints (e.g., multipoint connection), a specific function is needed within the data link layer to identify the data-link-connections using such a physical-connection.

7.6.4.2 *Data-link-service-data-unit mapping*

This function maps data-link-service-data-units into data-link-protocol-data-units on a one-to-one basis.

Note – More general mappings are for further study.

7.6.4.3 *Data-link-connection splitting*

This function performs splitting of one data-link-connection onto several physical-connections.

7.6.4.4 *Delimiting and synchronization*

These functions provide recognition of a sequence of physical-service-data-units (i.e., bits, see § 7.7.3.2) transmitted over the physical-connection, as a data-link-protocol-data-unit.

Note – These functions are sometimes referred to as framing.

7.6.4.5 *Sequence control*

This function maintains the sequential order of data-link-service-data-units across a data-link-connection.

7.6.4.6 *Error detection*

This function detects transmission, format and operational errors occurring either on the physical-connection, or as a result of a malfunction of the correspondent data-link-entity.

7.6.4.7 *Error recovery*

This function attempts to recover from detected transmission, format and operational errors and notifies the network-entities of errors which are unrecoverable.

7.6.4.8 *Flow control*

This function provides the flow control service as indicated in § 7.6.3.6.

7.6.4.9 *Identification and parameter exchange*

This function performs data-link-entity identification and parameter exchange.

7.6.4.10 *Control of data-circuit interconnection*

This function conveys to network-entities the capability of controlling the interconnection of data-circuits within the physical layer.

7.6.4.11 *Data link layer management*

The data link layer protocols deal with some management activities of the layer (such as activation and error control). See § 5.9 for the relationship with other management aspects.

7.7 *Physical layer*

7.7.1 *Definitions*

7.7.1.1 **data-circuit**

A communication path in the physical media for OSI between two physical-entities, together with the facilities necessary in the physical layer for the transmission of bits on it.

7.7.2 *Purpose*

The physical layer provides mechanical, electrical, functional and procedural means to activate, maintain and deactivate physical-connections for bit transmission between data-link-entities. A physical-connection may involve intermediate open systems, each relaying bit transmission within the physical layer. Physical layer entities are interconnected by means of a physical medium.

7.7.3 *Services provided to the data link layer*

The following services or elements of services provided by the physical layer are described below:

a) physical-connections;
b) physical-service-data-units;
c) physical-connection-endpoints;
d) data-circuit identification;
e) sequencing;
f) fault condition notification; and
g) quality of service parameters.

7.7.3.1 *Physical-connections*

The physical layer provide for the transparent transmission of bit streams between data-link-entities across physical-connections.

A data-circuit is a communication path in the physical media for OSI between two physical-entities, together with the facilities necessary in the physical layer for the transmission of bits on it.

A physical-connection may be provided by the interconnection of data-circuits using relaying functions in the physical layer. The provision of a physical-connection by such an assembly of data-circuits is illustrated in Figure 21/X.200.

The control of the interconnection of data-circuits is offered as a service to data-link-entities.

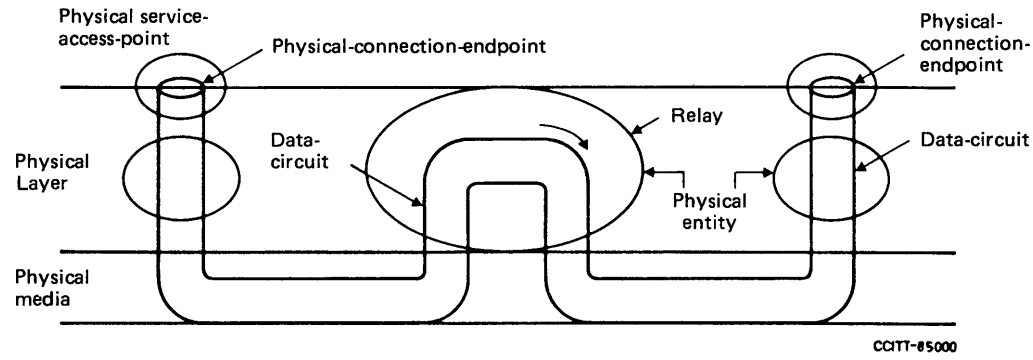


FIGURE 21/X.200

Interconnection of data-circuits within the physical layer

7.7.3.2 *Physical-service-data-units*

A physical-service-data-unit consists of one bit in serial transmission and of "n" bits in parallel transmission.

A physical-connection may allow duplex or half-duplex transmission of bit streams.

7.7.3.3 *Physical-connection-endpoints*

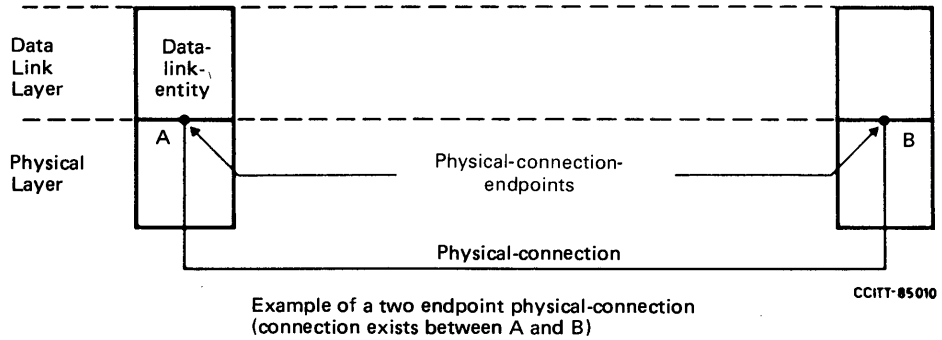
The physical layer provides physical-connection-endpoint-identifiers which may be used by a data-link-entity to identify physical-connection-endpoints.

A physical-connection will have two (point-to-point) or more (multi-endpoint) physical-connection-endpoints (see Figure 22/X.200).

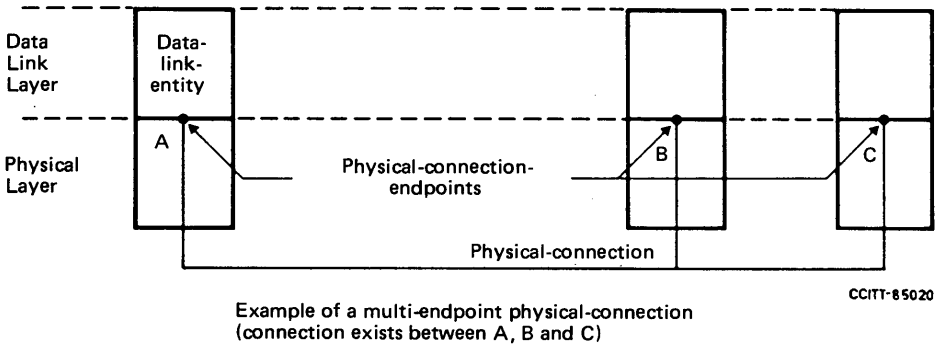
7.7.3.4 *Data-circuit identification*

The physical layer provides identifiers which uniquely specify the data-circuits between two adjacent open systems.

Note – This identifier is used by network-entities in adjacent open systems to refer to data-circuits in their dialogue.



Example of a two endpoint physical-connection
(connection exists between A and B)



Example of a multi-endpoint physical-connection
(connection exists between A, B and C)

FIGURE 22/X.200

Examples of physical connections

7.7.3.5 *Sequencing*

The physical layer delivers bits in the same order in which they were submitted.

7.7.3.6 *Fault condition notification*

Data-link-entities are notified or fault conditions detected within the physical layer.

7.7.3.7 *Quality of service parameters*

The quality of service of a physical-connection is derived from the data-circuits forming it. The quality of service can be characterized by:

a) error rate, where errors may arise from alteration, loss, creation, and other causes;
b) service availability;
c) transmission rate; and
d) transit delay.

7.7.4 *Functions within the physical layer*

The following functions performed by the physical layer are described below:

a) physical-connection activation and deactivation;

b) physical-service-data-unit transmission; and

c) physical layer management.

7.7.4.1 *Physical-connection activation and deactivation*

These functions provide for the activation and deactivation of physical-connections between two data-link-entities upon request from the data link layer. These include a relay function which provides for interconnection of data-circuits.

7.7.4.2 *Physical-service-data-unit transmission*

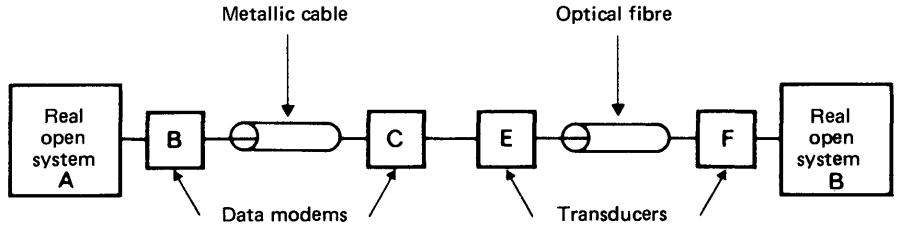
The transmission of physical-service-data-units (i.e., bits) may be synchronous or asynchronous.

7.7.4.3 *Physical layer management*

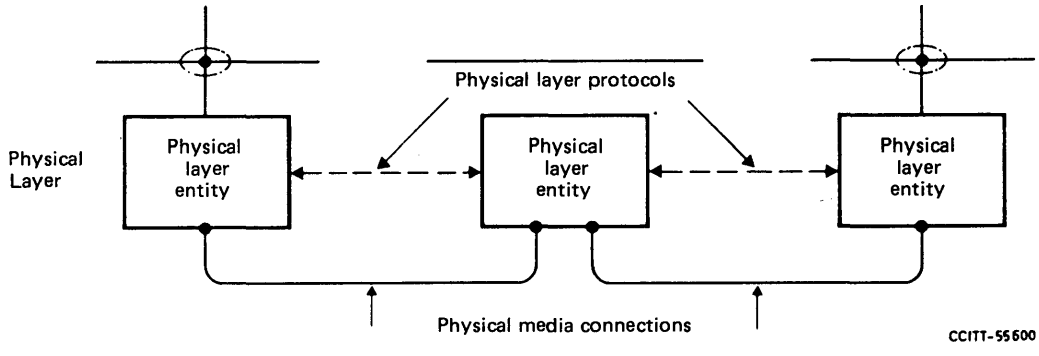
The physical layer protocols deal with some management activities of the layer (such as activation and error control). See sub-division 5.9 for the relationship with other management aspects.

Note – Relationship of the physical layer with the real environment

The above text deals with interconnection between open systems as illustrated in Figure 12/X.200. For open-systems to communicate in the real environment, real physical connections should be made, for example as in Figure 23a/X.200. Their logical representation is as shown in Figure 23b/X.200 and is called the physical media connection.



a) Real environment



b) Logical environment

Note – The area of physical media connections in OSI requires further study.

FIGURE 23/X.200

Examples of interconnection

The mechanical, electromagnetic and other media dependant characteristics of physical media connections are defined at the boundary between the physical layer and the physical media. Definitions of such characteristics may be found in other Recommendations.

ANNEX A

(to Recommendation X.200)

Brief explanation of how the layers were chosen

This Annex provides elements giving additional information to this Recommendation, which are not an integral part of it.

The following is a brief explanation of how the layers were chosen:

a) It is essential that the architecture permit usage of a realistic variety of physical media for interconnection with different control procedures (e.g., V.24, V.25, etc.). Application of principles P3, P5 and P8 leads to identification of a *Physical Layer* as the lowest layer in the architecture.

b) Some physical communication media (e.g. telephone line) require specific techniques to be used in order to transmit data between systems despite a relatively high error rate (i.e., an error rate not acceptable for the great majority of applications). These specific techniques are used in data-link control procedures which have been studied and standardized for a number of years. It must also be recognized that new physical communication media (e.g., fibre optics) will require different data link control procedures. Application of principles P3, P5 and P8 leads to identification of a *Data Link Layer* on top of the Physical Layer in the architecture.

c) In the open system architecture, some open systems will act as the final destination of data, see division 4. Some open systems may act only as intermediate nodes (forwarding data to other open systems), see Figure 13/X.200. Application of principles P3, P5 and P7 leads to identification of a *Network Layer* on top of the Data Link Layer. Network oriented protocols such as routing, for example, will be grouped in this layer. Thus, the Network Layer will provide a connection path (network-connection) between a pair of transport-entities, including the case where intermediate nodes are involved, see Figure 13/X.200 (see also § 7.5.4.1).

d) Control data transportation from source end open system to destination end open system (which is not performed in intermediate nodes) is the last function to be performed in order to provide the totality of the transport-service. Thus, the upper layer in the transport-service part of the architecture is the *Transport Layer*, on top of the Network Layer. This Transport Layer relieves higher layer entities from any concern with the transportation of data between them.

e) There is a need to organize and synchronize dialogue, and to manage the exchange of data. Application of principles P3 and P4 leads to the identification of a *Session Layer* on top of the Transport Layer.

f) The remaining set of general interest functions are those related to representation and manipulation of structured data for the benefit of application programs. Application of principles P3 and P4 leads to identification of a *Presentation Layer* on top of the Session Layer.

g) Finally, there are applications consisting of application processes which perform information processing. An aspect of these application processes and the protocols by which they communicate comprise the *Application Layer* as the highest layer of the architecture.

The resulting architecture with seven layers, illustrated in Figure 12/X.200 obeys principles P1 and P2.

A more detailed definition of each of the seven layers identified above is given in division 7 of this Recommendation, starting from the top with the Application Layer described in § 7.1 down to the Physical Layer described in § 7.7.

ANNEX B

(to Recommendation X.200)

Alphabetical index to definitions

Recommendation X.208

SPECIFICATION OF ABSTRACT SYNTAX NOTATION ONE (ASN.1)[1]

(Melbourne, 1988)

The CCITT,

considering

(a) the variety and complexity of information objects conveyed within the application layer;

(b) the need for a high-level notation for specifying such information objects;

(c) the value of isolating and standardizing the rules for encoding such information objects,

unanimously recommends

(1) that the notation for defining the abstract syntax of information objects is defined in Section 1;

(2) that character string types are defined in Section 2;

(3) that other useful types are defined in Section 3;

(4) that subtypes are defined in Section 4.

CONTENTS

[1] Recommendation X.208 and ISO 8824 [Information processing systems – Open Systems Interconnection – Specification of Abstract Syntax Notation One (ASN.1)] as extended by Addendum 1 to ISO 8824, were developed in close cooperation and are technically aligned.

0 Introduction

In the lower layers of the Basic Reference Model (see Recommendation X.200), each user data parameter of a service primitive is specified as the binary value of a sequence octets.

In the presentation layer, the nature of user data parameters changes. Application layer specifications require the presentation service user data (see Recommendation X.216) to carry the value of quite complex types, possibly including strings of characters from a variety of character sets. In order to specify the value which is carried, they require a defined notation which does not determine the representation of the value. This is supplemented by the specification of one or more algorithms called **encoding rules** which determine the value of the session layer octets carrying such application layer values (called the **transfer syntax**). The presentation layer protocol (see Recommendation X.226) can negotiate which transfer syntaxes are to be used.

The purpose of specifying a value is to distinguish it from other possible values. The collection of the value together with the values from which it is distinguished is called a **type**, and one specific instance is a value of that type. More generally, a value or type can often be considered as composed of several simpler values or types, together with the relationships between them. The term datatype is often used as a synonym for type.

In order to correctly interpret the representation of a value (whether by marks on paper or bits on communication line), it is necessary to know (usually from the context), the type of the value being represented. Thus the identification of a type is an important part of this Recommendation.

A very general technique for defining a complicated type is to define a small number of **simple types** by defining all possible values of the simple types, then combining these simple types in various ways. Some of the ways of defining new types are as follows:

a) given an (ordered) list of existing types, a value can be formed as an (ordered) sequence of values, one from each of the existing types; the collection of all possible values obtained in this way is a new type; (if the existing types in the list are all distinct, this mechanism can be extended to allow omission of some values from the list);

b) given a list of (distinct) existing types, a value can be formed as an (unordered) set of values, one from each of the existing types; the collection of all possible values obtained in this way is a new type; (the mechanism can again be extended to allow omission of some values);

c) given a single existing type, a value can be formed (ordered) sequence or (unordered) set of zero, one or more values of the existing type; the (infinite) collection of all possible values obtained in this way is a new type;

d) given a list of (distinct) types, a value can be chosen from any one of them; the set of all possible values obtained in this way is a new type;

e) given a type, a new type can be formed as a subset of it by using some structure or order relationship among the values;

Types which are defined in this way are called **structured types**.

Every type defined using the notation specified in this Recommendation is assigned a **tag**. The tag is defined either by this Recommendation or by the user of the notation.

It is common for the same tag to be assigned to many different types, the particular type being identified by the context in which the tag is used.

The user of the notation may choose to assign distinct tags to two occurrences of a single type, thereby creating two distinct types. This can be necessary when it is required to distinguish which choice has been made in situations such as d) above.

Four classes of tag are specified in the notation.

The first is the **universal** class. Universal class tags are only used as specified within this Recommendation, and each tag is either

a) assigned to a single type; or

b) assigned to a construction mechanism.

The second class of tag is the **application** class. Application class tags are assigned to types by other standards or Recommendations. Within a particular standard or Recommendation, an application class tag is assigned to only one type.

The third class is the **private** class. Private class tags are never assigned by ISO Standards or CCITT Recommendations. Their use is enterprise specific.

The final class of tag is the **context-specific** class. This is freely assigned within any use of this notation, and is interpreted according to the context in which it is used.

Tags are mainly intended for machine use, and are not essential for the human notation defined in this Recommendation. Where, however, it is necessary to require that certain types be distinct, this is expressed by requiring that they have distinct tags. The allocation of tags is therefore an important part of the use of this notation.

Note 1 – All types which can be defined in the notation of this Recommendation have a tag. Given any type, the user of the notation can define a new type with a different tag.

Note 2 – Encoding rules always carry the tag of a type, explicitly or implicitly, with any representation of a value of the type. The restrictions placed on the use of the notation are designed to ensure that the tag is sufficient to unambiguously determine the actual type, provided the applicable type of definitions are available.

This Recommendation specifies a notation which both enables complicated types to be defined and also enables values of these types to be specified. This is done without determining the way an instance of this type is to be represented (by a sequence of octets) during transfer. A notation which provides this facility is called a **notation for abstract syntax definition.**

The purpose of this Recommendation is to specify a notation for abstract syntax definition called **Abstract Syntax Notation One**, or ASN.1. Abstract Syntax Notation One is used as a semi-formal tool to define protocols. The use of the notation does not necessarily preclude ambiguous specifications. It is the responsibility of the users of the notation to ensure that their specifications are not ambiguous.

This Recommendation is supported by other standards and Recommendations which specify encoding rules. The application of encoding rules to the value of a type defined by ASN.1 results in a complete specification of the representation of values of that type during transfer (a transfer syntax).

This Recommendation is technically and editorially aligned with ISO 8824 plus Addendum 1 to ISO 8824.

Section one of this Recommendation defines the simple types supported by ASN.1, and specifies the notation to be used for referencing simple types and defining structured types. Section one also specifies the notation to be used for specifying values of types defined using ASN.1.

Section two of this Recommendation defines additional types (character string types) which, by the application of encoding rules for character sets, can be equated with the octetstring type.

Section three of this Recommendation defines certain structured types which are considered to be of general utility, but which require no additional encoding rules.

Section four of this Recommendation defines a notation which enables subtypes to be defined from the values of a parent type.

Annex A is part of this Recommendation, and specifies a notation for extending the basic ASN.1 notation. This is called the macro facility.

Annex B is part of this Recommendation, and defines the object identifier tree for authorities supported by ISO.

Annex C is part of this Recommendation and defines the object identifier tree for authorities supported by CCITT.

Annex D is part of this Recommendation and defines the object identifier tree for joint use by ISO and CCITT.

Appendix I is not part of this Recommendation, and provides examples and hints on the use of the ASN.1 notation.

Appendix II is not part of this Recommendation, and provides a summary of ASN.1 using the notation of § 5.

The text of this Recommendation, and in particular the annexes B to D, are the subject of joint ISO-CCITT agreement.

1 Scope and field of application

This Recommendation specifies a notation for abstract syntax definition called Abstract Syntax Notation One (ASN.1).

This Recommendation defines a number of simple types, with their tags, and specifies a notation for referencing these types and for specifying values of these types.

This Recommendation defines mechanisms for constructing new types from more basic types, and specifies a notation for defining such structured types and assigning them tags, and for specifying values of these types.

This Recommendation defines character sets for use within ASN.1.

This Recommendation defines a number of useful types (using ASN.1), which can be referenced by users of ASN.1.

The ASN.1 notation can be applied whenever it is necessary to define the abstract syntax of information. It is particularly, but not exclusively, applicable to application protocols.

The ASN.1 notation is also referenced by other presentation layer standards and Recommendations which define encoding rules for the simple types, the structured types, the character string types and the useful types defined in ASN.1.

2 References

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

[2] Recommendation X.209, *Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1)* (see also ISO 8825).

[3] Recommendation X.216, (see also ISO 8822), *Presentation Service Definition for Open Systems Interconnection for CCITT Applications.*

[4] Recommendation X.226, *Presentation Protocol Specification for Open Systems Interconnection for CCITT Applications* (see also ISO 8823).

[5] ISO 2014, *Writing of calendar dates in all-numeric form.*

[6] ISO 2375, *Data processing – Procedure for registration of escape sequences.*

[7] ISO 3166, *Codes for the representation of names of countries.*

[8] ISO 3307, *Information interchange – Representations of time of the day.*

[9] ISO 4031, *Information interchange – Representation of local time differentials.*

[10] ISO 6523, *Data interchange – Structure for identification of organizations.*

[11] Recommendation X.121, *International numbering plan for public data networks.*

3 Definitions

The definitions in Recommendation X.200 are used in this Recommendation.

3.1 value

A distinguished member of a set of values.

3.2 **type**

A named set of values.

3.3 **simple type**

A type defined by directly specifying the set of its values.

3.4 **structured type**

A type defined by reference to one or more other types.

3.5 **component type**

One of the types referenced when defining a structured type.

3.6 **tag**

A type denotation which is associated with every ASN.1 type.

3.7 **tagging**

Replacing the existing (possibly the default) tag of a type by a specified tag.

3.8 **ASN.1 character set**

The set of characters, specified in § 7, used in the ASN.1 notation.

3.9 **items**

Named sequences of characters from the ASN.1 character set, specified in § 8, which are used to form the ASN.1 notation.

3.10 **type (or value) reference name**

A name associated uniquely with a type (or value) within some context.

Note – Reference names are assigned to the types defined in this Recommendation; these are universally available within ASN.1. Other reference names are defined in other standards and Recommendations, and are applicable only in the context of the standard or Recommendation.

3.11 **ASN.1 encoding rules**

Rules which specify the representation during transfer of the value of any ASN.1 type; ASN.1 encoding rules enable information being transferred to be identified by the recipient as a specific value of a specific ASN.1 type.

3.12 **character string type**

A type whose values are strings of characters from some defined character set.

3.13 **Boolean type**

A simple type with two distinguished values.

3.14 **true**

One of the distinguished values of the Boolean type.

3.15 **false**

The other distinguished value of the Boolean type.

3.16 **integer type**

A simple type with distinguished values which are the positive and negative whole numbers, including zero (as a single value).

Note – Particular encoding rules limit the range of an integer, but such limitations are chosen so as not to affect any user of ASN.1.

3.17 **enumerated type**

A simple type whose values are given distinct identifiers as part of the type notation.

3.18 **real type**

A simple type whose distinguished values (specified in § 16.2) are members of the set of real numbers.

3.19 **bitstring type**

A simple type whose distinguished values are an ordered sequence of zero, one or more bits.

Note – Encoding rules do not limit the number of bits in a bit-string.

3.20 **octetstring type**

A simple type whose distinguished values are an ordered sequence of zero, one or more octets, each octet being an ordered sequence of eight bits.

Note – Encoding rules do not limit the number of octets in an octet string.

3.21 **null type**

A simple type consisting of a single value, also called null.

Note – The null value is commonly used where several alternatives are possible, but none of them apply.

3.22 **sequence type**

A structured type, defined by referencing a fixed, ordered, list of types (some of which may be declared to be optional); each value of the new type is an ordered list of values, one from each component type.

Note – Where a component type is declared to be optional, a value of the new type need not contain a value of that component type.

3.23 **sequence-of type**

A structured type, defined by referencing a single existing type; each value in the new type is an ordered list of zero, one or more values of the existing type.

Note – Encoding rules do not limit the number of values in a sequence-of value.

3.24 **set type**

A structured type, defined by referencing a fixed, unordered, list of distinct types (some of which may be declared to be optional); each value in the new type is an unordered list of values, one from each of the component types.

Note – Where a component type is declared to be optional, the new type need not contain a value of that component type.

3.25 **set-of type**

A structured type, defined by referencing a single existing type; each value in the new type is an unordered list zero, one or more values of the existing type.

Note – Encoding rules do not limit the number of values in a set-of value.

3.26 **tagged type**

A type defined by referencing a single existing type and a tag; the new type is isomorphic to the existing type, but is distinct from it.

3.27 **choice type**

A structured type, defined by referencing a fixed, unordered, list of distinct types; each value of the new type is a value of one of the component types.

3.28 **selection type**

A structured type, defined by reference to a component type of a choice type.

3.29 **any type**

A choice type whose component types are unspecified, but are restricted to the set of types which can be defined using ASN.1.

3.30 **external type**

A type whose distinguished values cannot be deduced from their characterisation as external, but which can be deduced from the encoding of such a value; the value may, but need not, be describable using ASN.1, and thus their encoding may, but need not, conform to ASN.1 encoding rules.

3.31 **information object**

A well-defined piece of information, definition, or specification which requires a name in order to identify its use in an instance of communication.

3.32 **object identifier**

A value (distinguishable from all other such values) which is associated with an information object.

3.33 **object identifier type**

A type whose distinguished values are the set of all object identifiers allocated in accordance with the rules of this Recommendation.

Note – The rules of this Recommendation permit a wide range of authorities to independently associate object identifiers with information objects.

3.34 **object descriptor type**

A type whose distinguished values are human-readable text providing a brief description of an information object.

Note – An object descriptor value is usually, but not always associated with a single information object. Only an object identifier value unambiguously identifies an information object.

3.35 **recursive definitions**

A set of ASN.1 definitions which cannot be reordered so that all types used in a construction are defined before the definition of the construction.

Note – Recursive definitions are allowed in ASN.1: the user of the notation has the responsibility of ensuring that those values (of the resulting types) which are used have a finite representation.

3.36 **module**

One or more instances of the use of the ASN.1 notation for type and value definition, encapsulated using the ASN.1 module notation (see § 9).

3.37 **production**

A part of the formal notation used to specify ASN.1, in which allowed sequences of items are associated with a name which can be used to reference those sequences in the definition of new sets of allowed sequences.

3.38 **Coordinated Universal Time (UTC)**

The time scale maintained by the Bureau Internationale de l'Heure (International Time Bureau) that forms the basis of a coordinated dissemination of standard frequencies and time signals.

Note 1 – The source of this definition is Recommendation 460-2 of the Consultative Committee on International Radio (CCIR). CCIR has also defined the acronym for Coordinated Universal Time as UTC.

Note 2 – UTC is also referred to as Greenwich Mean Time and appropriate time signals are regularly broadcast.

3.39 **user (of ASN.1)**

The individual or organization that defines the abstract syntax of a particular piece of information using ASN.1.

3.40 **subtype (of a parent type)**

A type whose values are specified as a subset of the values of some other type (the parent type).

3.41 **parent type (of a subtype)**

Type used to define a subtype.

Note – The parent type may itself be a subtype of some other type.

3.42 **subtype specification**

A notation which can be used in association with the notation for a type, to define a subtype of that type.

3.43 **subtype value set**

A notation forming part of a subtype specification, specifying a set of values of the parent type which are to be included in the subtype.

3.44 This Recommendation uses the following terms defined in Recommendation X.216:

a) presentation data value; and
b) (an) abstract syntax; and
c) abstract syntax name; and
d) transfer syntax name.

3.45 This Recommendation also uses the following terms defined in ISO 6523;

a) issuing organization; and
b) organization code; and
c) International Code Designator.

3.46 This Recommendation uses the following term defined in Recommendation X.226:

a) presentation context identifier.

4 Abbreviations

ASN.1 Abstract Syntax Notation One.
UTC Coordinated Universal Time.
ICD International Code Designator.
DCC Data Country Code
DNIC Data Network Identification Code.

5 Notation used in this Recommendation

The ASN.1 notation consists of a sequence of characters from the ASN.1 character set specified in § 7.

Each use of the ASN.1 notation contains characters from the ASN.1 character set grouped into items. Clause 8 specifies all the sequences of characters forming ASN.1 items, and names each item.

The ASN.1 notation is specified in § 9 (and following clauses) by specifying the collection of sequences of items which form valid instances of the ASN.1 notation, and by specifying the semantics of each sequence.

In order to specify these collections, this Recommendation uses a formal notation defined in the following sub-clauses.

5.1 *Productions*

A new (more complex) collection of ASN.1 sequences is defined by means of a production. This uses the names of collections of sequences defined in this Recommendation and forms a new collection of sequences by specifying either

a) that the new collection of sequences is to consist of any of the original collections; or

b) that the new collection is to consist of any sequence which can be generated by taking exactly one sequence from each collection, and juxtaposing them in a specified order.

Each production consists of the following parts, on one or several lines, in order:

a) a name for the new collection of sequences;

b) the characters

::=

c) one or more alternative collections of sequences, defined as in § 5.2, separated by the character

|

A sequence is present in the new collection if it is present in one or more of the alternative collections. The new collection is referenced in this Recommendation by the name in a) above.

Note – If the same sequence appears in more than one alternative, any semantic ambiguity in the resulting notation is resolved by other parts of the complete ASN.1 sequence.

5.2 *The alternative collections*

Each of the alternative collections of sequences in "one or more alternative collections of" is specified by a list of names. Each name is either the name of an item, or is the name of a collection of sequences defined by a production in this Recommendation.

The collection of sequences defined by the alternative consists of all sequences obtained by taking any one of the sequences (or the item) associated with the first name, in combination with (and followed by) any one of the sequences (or item) associated with the second name, in combination with (and followed by) any one of the sequences (or item) associated with the third name, and so on up to and including the last name (or item) in the alternative.

5.3 *Example of a production*

```
BitStringValue ::=
        bstring |
        hstring |
        {IdentifierList}
```

is a production which associates with the name **BitStringValue** the following sequences:

a) any bstring (an item); and

b) any hstring (an item); and

c) any sequence associated with **IdentifierList**, preceded by a { and followed by a }

Note – { and } are the names of items containing the single characters { and } (see § 8).

In this example, **IdentifierList** would be defined by a further production, either before or after production defining **BitStringValue**.

5.4 *Layout*

Each production used in this Recommendation is preceded and followed by an empty line. Empty lines do not appear within productions. The production may be on a single line, or may be spread over several lines. Layout is not significant.

5.5 *Recursion*

The productions in this Recommendation are frequently recursive. In this case the productions are to be continously reapplied until no new sequences are generated.

Note – In many cases, such reapplication results in an unbounded collection of allowed sequences, some or all of which may themselves be unbounded. This is not an error.

5.6 *References to a collection of sequences*

This Recommendation references a collection of sequences (part of the ASN.1 notation) by referencing the first name (before the ::=) in a production; the name is surrounded by " to distinguish it from natural language text, unless it appears as part of a production.

5.7 *References to an item*

This Recommendation references an item by referencing the name of the item; the name is surrounded by " to distinguish it from natural language text, unless it appears as part of a production.

TABLE 1/X.208

Universal class tag assignments

UNIVERSAL 1	Boolean type
UNIVERSAL 2	Integer type
UNIVERSAL 3	Bitstring type
UNIVERSAL 4	Octetstring type
UNIVERSAL 5	Null type
UNIVERSAL 6	Object identifier type
UNIVERSAL 7	Object descriptor type
UNIVERSAL 8	External type
UNIVERSAL 9	Real type
UNIVERSAL 10	Enumerated type
UNIVERSAL 12 à 15	Reserved for future versions of this Recommendation
UNIVERSAL 16	Sequence and Sequence-of types
UNIVERSAL 17	Set and Set-of types
UNIVERSAL 18-22, 25-27	Character string types
UNIVERSAL 23-24	Time types
UNIVERSAL 28-...	Reserved for future versions of this Recommendation

5.8 *Tags*

A tag is specified by giving its class and the number within the class. The class is one of

universal
application
private
context-specific

The number is a non-negative integer, specified in decimal notation.

Restrictions on tags assigned by the user of ASN.1 are specified in § 26.

Tags in the universal class are assigned in such a way that, for structured types, the top-level structure can be deduced from the tag, and for simple types, the type can be deduced from the tag. Table 1/X.208 summarises the assignment of tags in the universal class which are specified in this Recommendation.

Note – Additional tags in the universal class are reserved for assignment by future versions of this Recommendation.

6 Use of the ASN.1 notation

6.1 The ASN.1 notation for a type definition shall be "Type" (see § 12.1).

6.2 The ASN.1 notation for a value of a type shall be "Value" (see § 12.7).

Note – It is not in general possible to interpret the value notation without knowledge of the type.

6.3 The ASN.1 notation for assigning a type to a type reference name shall be "Typeassignment" (see § 11.1).

6.4 The ASN.1 notation for assigning a value reference name shall be "Valueassignment" (see § 11.2).

6.5 The notation "Typeassignment" and "Valueassignment" shall only be used within the notation "Module-Definition" (but see § 9.1).

SECTION 1 – SPECIFICATION OF ASN.1 NOTATION

7 The ASN.1 character set

7.1 An ASN.1 item shall consist of a sequence of the characters listed in Table 2/X.208, except as specified in § 7.2 and § 7.3.

TABLE 2/X.208

ASN.1 characters

A to Z
a to z
0 to 9
: = , { } < .
( ) [ ] - ' "

Note 1 – The additional characters > and | are used in the macro-notation (see Annex A).

Note 2 – Where equivalent derivative standards are develped by national standards bodies, additional characters may appear in the following items (the last five of which are defined in Annex A):

typereference	(§ 8.2.1)
identifier	(§ 8.3)
Valuereference	(§ 8.4)
modulereference	(§ 8.5)
macroreference	(§ A.2.1)
productionreference	(§ A.2.2)
localtypereference	(§ A.2.3)
localvaluereference	(§ A.2.4)
astring	(§ A.2.7)

Note 3 – When additional characters are introduced to accommodate a language in which the distinction between upper-case and lower-case letters is without meaning, the syntactic distinction achieved by dictating the case of the first character of certain of the above ASN.1 items has to be achieved in some other way.

7.2 Where the notation used to specify the value of a character string type, all characters of the defined character set can appear in the ASN.1 notation, surrounded by the characters ". (See § 8.11.)

7.3 Additional characters may appear in the "comment" item. (See § 8.6.)

7.4 There shall be no significance placed on the typographical style, size, colour, intensity, or other display characteristics.

7.5 The upper and lower case letters shall be regarded as distinct.

8 ASN.1 items

8.1 *General rules*

8.1.1 The following subclauses specify the characters in ASN.1 items. In each case the name of the item is given, together with the definition of the character sequences which form the item.

Note – Annex A specifies additional items used in the macro notation.

8.1.2 Each item specified in the following subclauses shall appear on a single line, and (except for the "comment" item) shall not contain spaces.

8.1.3 The length of a line is not restricted.

8.1.4 The items in the sequences specified by this Recommendation (the ASN.1 notation) may appear on one line or may appear on several lines, and may be separated by one or more spaces or empty lines.

8.1.5 An item shall be separated from a following item by a space, or by being placed on a separate line, if the initial character (or characters) of the following item is a permitted character (or characters) for inclusion at the end of the characters in the earlier item.

8.2 *Type references*

Name of item-typereference.

8.2.1 A "typereference" shall consist of an arbitrary number (one or more) of letters, digits, and hyphens. The initial character shall be an upper-case letter. A hyphen shall not be the last character. A hyphen shall not be immediately followed by another hyphen.

Note – The rules concerning hyphen are designed to avoid ambiguity with (possibly following) comment.

8.2.2 A "typereference" shall not be one of the reserved character sequences listed in Table 3/X.208.

Note – § A.2.9 specifies additional reserved character sequences when within a macro definition.

8.3 *Identifiers*

Name of item-identifier.

An "identifier" shall consist of an arbitrary number (one or more) of letters, digits, and hyphens. The initial character shall be a lower-case letter. A hyphen shall not be the last character. A hyphen shall not be immediately followed by another hyphen.

Note – The rules concerning hyphen are designed to avoid ambiguity with (possibly following) comment.

TABLE 3/X.208

Reserved character sequences

BOOLEAN	BEGIN
INTEGER	END
BIT	DEFINITIONS
STRING	EXPLICIT
OCTET	ENUMERATED
NULL	EXPORTS
SEQUENCE	IMPORTS
OF	REAL
SET	INCLUDES
IMPLICIT	MIN
CHOICE	MAX
ANY	SIZE
EXTERNAL	FROM
OBJECT	WITH
IDENTIFIER	COMPONENT
OPTIONAL	PRESENT
DEFAULT	ABSENT
COMPONENTS	DEFINED
UNIVERSAL	BY
APPLICATION	PLUS-INFINITY
PRIVATE	MINUS-INFINITY
TRUE	TAGS
FALSE	

8.4 *Value references*

Name of item-valuereference.

A "valuereference" shall consist of the sequence of characters specified for an "identifier" in § 8.3. In analysing an instance of use of this notation, a "valuereference" is distinguished from an "identifier" by the context in which it appears.

8.5 *Module reference*

Name of item-modulereference.

A "modulereference" shall consist of the sequence of characters specified for a "typereference" in § 8.2. In analysing an instance of use of this notation, a "modulereference" is distinguished from a "typereference" by the context in which it appears.

8.6 *Comment*

Name of item-comment.

8.6.1 A "comment" is not referenced in the definition of the ASN.1 notation. It may, however, appear at any time between other ASN.1 items, and has no significance.

8.6.2 A "comment" shall commence with a pair of adjacent hyphens and shall end with the next pair of adjacent hyphens or at the end of the line, whichever occurs first. A comment shall not contain a pair of adjacent hyphens other than the pair which opens it and the pair, if any, which ends it. It may include characters which are not in the character set specified in § 7.1 (see § 7.3).

8.7 *Empty item*

Name of item-empty.

The "empty" item contains no characters. It is used in the notation of § 5 when alternative sets of sequences are specified, to indicate that absence of all aternatives is possible.

8.8 *Number item*

Name of item-number.

A "number" shall consist of one or more digits. The first digit shall not be zero unless the "number" is a single digit.

8.9 *Binary string item*

Name of item-bstring.

A "bstring" shall consist of an arbitrary number (possibly zero) of zeros and ones, preceded by a single ' and followed by the pair of characters:

'B

Example: '01101100'B

8.10 *Hexadecimal string item*

Name of item-hstring.

8.10.1 An "hstring" shall consist of an arbitrary number (possibly zero) of the characters

A B C D E F 0 1 2 3 4 5 6 7 8 9

preceded by a single ' and followed by the pair of characters

'H

Example: 'AB0196'H

8.10.2 Each character is used to denote the value of a semi-octet using a hexadecimal representation.

8.11 *Character string item*

Name of item-cstring.

A "cstring" shall consist of an arbitrary number (possibly zero) of characters from the character set referenced by a character string type, preceded and followed by ". If the character set includes the character ", this character shall be represented in the "cstring" by a pair of ". The character set involved is not limited to the character set listed in Table 2/X.208, but is determined by the type for which the "cstring" is a value (see § 7.2).

Example: "宛.姐 虻 飴 絢"

8.12 *Assignment item*

Name of item-"::="

This item shall consist of the sequence of characters

::=

Note – This sequence does not contain any space characters (see § 8.1.2).

8.13 *Single character items*

Names of items-

{
}
<
,
.
(
)
[
]
- (hyphen)
;

An item with any of the names listed above shall consist of the single character forming the name.

8.14 *Keyword items*

Names of items –

BOOLEAN
INTEGER
BIT
STRING
OCTET
NULL
SEQUENCE
OF
SET
IMPLICIT
CHOICE
ANY
EXTERNAL
OBJECT
IDENTIFIER
OPTIONAL
DEFAULT
COMPONENTS
UNIVERSAL
APPLICATION
PRIVATE
TRUE
FALSE
BEGIN
END
DEFINITIONS
EXPLICIT
ENUMERATED
EXPORTS
IMPORTS
REAL
INCLUDES
MIN
MAX
SIZE
FROM

WITH
COMPONENT
PRESENT
ABSENT
DEFINED
BY
PLUS-INFINITY
MINUS-INFINITY
TAGS

Items with the above names shall consist of the sequence of characters in the name.

Note 1 – Spaces do not occur in these sequences.

Note 2 – Where these sequences are not listed as reserved sequences in § 8.2.2, they are distinguished from other items containing the same characters by the context in which they appear.

9 Module definition

9.1 A "ModuleDefinition" is specified by the following productions:

ModuleDefinition ::=

ModuleIdentifier
DEFINITIONS
TagDefault
"::="
BEGIN
ModuleBody
END

TagDefault ::=

EXPLICIT TAGS |
IMPLICIT TAGS |
empty

ModuleIdentifier ::=

modulereference
AssignedIdentifier

AssignedIdentifier ::=

ObjectIdentifierValue |
empty

ModuleBody ::=

Exports Imports AssignmentList |
empty

Exports ::=

EXPORTS SymbolsExported; |
empty

SymbolsExported ::=

SymbolList |
empty

Imports ::=

IMPORTS SymbolsImported; |
empty

SymbolsImported ::=

SymbolsFromModuleList |
empty

SymbolsFromModuleList ::=

SymbolsFromModule SymbolsFromModuleList |
SymbolsFromModule

SymbolsFromModule ::=

SymbolList FROM ModuleIdentifier

SymbolList ::= Symbol, SymbolList | Symbol

Symbol ::=typereference | valuereference

AssignmentList ::=

Assignment AssignmentList |
Assignment

Assignment ::=

TypeAssignment | ValueAssignment

Note 1 – Annex A specifies a "MacroDefinition" sequence which can also appear in the "AssignmentList". Notations defined by a macro definition may appear before or after the macro definition, within the same module.

Note 2 – In individual (but deprecated) cases, and for examples and for the definition of types with universal class tags, the "ModuleBody" can be used outside of a "ModuleDefinition".

Note 3 – "Typeassignment" and "Valueassignment" productions are specified in § 11.

Note 4 – The grouping of ASN.1 datatypes into modules does not necessarily determine the formation of presentation data values into named abstract syntaxes for the purpose of presentation context definition.

Note 5 – The value of "TagDefault" for the module definition affects only those types defined explicitly in the module. It does not affect the interpretation of imported types.

Note 6 – A "macroreference" (see Annex A), can also appear as a "Symbol".

9.2 The "TagDefault" is taken as "EXPLICIT TAGS" if it is "empty".

Note – § 26 gives the meaning of both "EXPLICIT TAGS" and "IMPLICIT TAGS".

9.3 The "modulereference" appearing in the "ModuleDefinition" production is called the module name. Module names are chosen so as to ensure consistency and completeness of all "Assignment" sequences appearing within the "ModuleBody" of all "ModuleDefinition" sequences with this module name. A set of "Assignment" sequences is consistent and complete if, for every "typereference" or "valuereference" appearing within it, there is exactly one "Typeassignment" or "Valueassignment" (respectively) associating the name with a type or value (respectively), or exactly one "SymbolsFromModule" in which the "typereference" or "valuereference" (respectively) appears as a "Symbol".

9.4 Module names should be used only once (except as specified in § 9.10) within the sphere of interest of the definition of the module.

Note – It is recommended that modules defined in ISO Standards should have module names of the form

ISOxxxx-yyyy

where xxxx is the number of the Standard, and yyyy is a suitable acronym for the Standard (e.g. JTM, FTAM, or CCR). A similar convention can be applied by other standards-making bodies.

9.5 If the "AssignedIdentifier" includes an "ObjectIdentifierValue", the latter unambiguously and uniquely identifies the module.

Note – It is recommended that an object identifier be assigned so that others can unambiguously refer to the module.

9.6 The "ModuleIdentifier" in a "SymbolsFromModule" shall appear in the "ModuleDefinition" of another module, except that if it includes an "ObjectIdentifierValue", the "modulereference" may differ in the two cases.

Note 1 – A different "modulereference" from that used in the other module should only be used when symbols are to be imported from two modules with the same name modules (the modules being named in disregard of clause 9.4). The use of alternative distinct names makes these names available for use in the body of the module (see § 9.8).

Note 2 – When both a "modulereference" and an "ObjectIdentifierValue" are used in referencing a module, the latter shall be considered definitive.

9.7 Each "SymbolsExported" shall be defined in the module being constructed.

Note – It is recommended that every symbol to which reference from outside the module is appropriate be included in the "SymbolsExported". If there are no such symbols, then the "empty" alternative of "SymbolsExported" (not of "Exports") should be selected.

9.8 Each "SymbolsFromModule" shall be defined in the module denoted by the "ModuleIdentifier" in "SymbolsFromModule". If "Exports" is used in the definition of that module, the "Symbol" shall appear in its "SymbolsExported".

9.9 A "Symbol" in a "SymbolsFromModule" may appear in "ModuleBody" in a "DefinedType" (if it is a "typereference") or "DefinedValue" (if it is a "valuereference"). The meaning associated with "Symbol" also appears in an "AssignmentList" (deprecated), or appears in one or more instances of "SymbolsFromModule", it shall only be used in a "ExternalTypeReference" or "ExternalValueReference" whose "modulereference" is that in "SymbolsFromModule" (see § 9.10). Where it does not so appear, it may be used in a "DefinedType" or "DefinedValue" directly.

9.10 Except as specified in § 9.9, a "typereference" or "valuereference" shall be referenced in a module different from that in which it is defined by using an "Externaltypereference" or "Externalvaluereference", specified by the following productions:

```
Externaltypereference ::=
        modulereference
        typereference

Externalvaluereference ::=
        modulereference
        valuereference
```

10 Referencing type and value definitions

10.1 The productions

```
DefinedType ::=
        Externaltypereference |
        typereference

DefinedValue ::=
        Externalvaluereference |
        valuereference
```

specify the sequences which shall be used to reference type and value definitions.

10.2 Except as specified in § 9.10, the "typereference" and "valuereference" alternatives shall not be used unless the reference is within the module in which a type is assigned (see § 11.1 and § 11.2) to the typereference or valuereference.

10.3 The "Externaltypereference" and "Externalvaluereference" shall not be used unless the corresponding "typereference" or "valuereference" has been assigned a type or value respectively (see § 11.1 and § 11.2) within the corresponding "modulereference".

11 Assigning types and values

11.1 A "typereference" shall be assigned a type by the notation specified by the "Typeassignment" production:

Typeassignment ::= typereference

"::="
Type

The "typereference" shall not be one of the names used to reference the character string types defined in section two, and shall not be one of the names used to reference the types defined in section three.

11.2 A "valuereference" shall be assigned a value by the notation specified by the "Valueassignment" production:

Valueassignment ::= valuereference

Type
"::="
Value

The "Value" being assigned to the "valuereference" shall be a valid notation (see § 12.7) for a value of the type defined by "Type".

12 Definition of types and values

12.1 A type shall be referenced by one of the sequences "Type":

Type ::= BuiltinType | DefinedType | Subtype
(see § 10.1) (see § 37)

BuiltinType ::=

BooleanType |
IntegerType |
BitStringType |
OctetStringType |
NullType |
SequenceType |
SequenceOfType |
SetType |
SetOfType |
ChoiceType |
SelectionType |
TaggedType |
AnyType |
ObjectIdentifierType |
CharacterStringType |
UsefulType |
EnumeratedType |
RealType |

Note 1 – A type notation defined in a macro can also be used as a sequence for "Type" (see Annex A).

Note 2 – Additional built-in types may be defined by future versions of this Recommendation.

12.2 The "BuiltinType" notation is specified in the following clauses.

12.3 The "Subtype" notation is specified in section four.

12.4 The type being referenced is the type defined by the "BuiltinType" or "Subtype" assigned to the "DefinedType".

12.5 In some notations within which a type is referenced, the type may be named. In such cases, this Recommendation specifies the use of the notation "NamedType":

NamedType ::=

identifier Type |
Type |
SelectionType

The notation "SelectionType" and the corresponding value notation is specified in § 25.

Note – The notation "SelectionType" contains an "identifier" which may form part of the value notation when "SelectionType" is used as a "NamedType" (see § 25.1).

12.6 The "identifier" is not part of the type, and has no effect on the type. The type referenced by a "NamedType" sequence is that referenced by the contained "Type" sequence.

12.7 The value of a type shall be specified by one of the sequences "Value":

Value ::= BuiltinValue | DefinedValue

BuiltinValue ::=

BooleanValue |
IntegerValue |
BitStringValue |
OctetStringValue |
NullValue |
SequenceValue |
SequenceOfValue |
SetValue |
SetOfValue |
ChoiceValue |
SelectionValue |
TaggedValue |
AnyValue |
ObjectIdentifierValue |
CharacterStringValue |
EnumeratedValue |
RealValue |

Note – A value notation defined in a macro may also be used as a sequence for "Value" (see Annex A).

12.8 If the type is defined using one of the notations shown on the left below, then the value shall be specified using the notation shown on the right below:

Type notation	**Value notation**
BooleanType	BooleanValue
IntegerType	IntegerValue
BitStringType	BitStringValue
OctetStringType	OctetStringValue
NullType	NullValue
SequenceType	SequenceValue
SequenceOfType	SequenceOfValue
SetType	SetValue
SetOfType	SetOfValue
ChoiceType	ChoiceValue
TaggedType	TaggedValue
AnyType	AnyValue
ObjectIdentifierType	ObjectIdentifierValue
CharacterStringType	CharacterStringValue
EnumeratedType	EnumeratedValue
RealType	RealValue

Note – Additional value notations may be defined by future versions of this Recommendation.

Where the type is a DefinedType, the value notation shall be the notation for a type used in producing the DefinedType.

12.9 The value notation for a type defined by the "UsefulType" notation is specified in section three.

12.10 The "BuiltinValue" notation is specified in the following clauses.

12.11 The value of a type referenced using the "NamedType" notation shall be defined by the notation "NamedValue":

```
NamedValue ::=
        identifier Value |
        Value
```

where the "identifier" (if any) is the same as that used in the "NamedType" notation. § 25.2 specifies further restrictions on the "NamedValue" when the "NamedType" was a "SelectionType".

Note – The "identifier" is part of the notation, it does not form part of the value itself.

12.12 The "identifier" shall be present in the "NamedValue" if and only if it was present in the "NamedType".

Note – An "identifier" is always present in the case of a "SelectionType".

13 Notation for the Boolean type

13.1 The Boolean type (see § 3.13) shall be referenced by the notation "BooleanType":

```
BooleanType ::= BOOLEAN
```

13.2 The tag for types defined by this notation is universal class, number 1.

13.3 The value of a Boolean type (see § 3.14 and § 3.15) shall be defined by the notation "BooleanValue":

```
BooleanValue ::= TRUE | FALSE
```

14 Notation for the integer type

14.1 The integer type (see § 3.16) shall be referenced by the notation "IntegerType":

```
IntegerType ::=
        INTEGER |
        INTEGER{NamedNumberList}
NamedNumberList ::=
        NamedNumber |
        NamedNumberList,NamedNumber
NamedNumber ::=
        identifier(SignedNumber) |
        identifier(DefinedValue)
SignedNumber ::= number | -number
```

14.2 The second alternative of "SignedNumber" shall not be used if the "number" is zero.

14.3 The "NamedNumberList" is not significant in the definition of a type. It is used solely in the value notation specified in § 14.9.

14.4 The "DefinedValue" shall be a reference to a value of type integer, or of a type derived from integer by tagging or subtyping.

14.5 The value of each "SignedNumber" or "DefinedValue" appearing in the "NamedNumberList" shall be different, and represents a distinguished value of the integer type.

14.6 Each "identifier" appearing in the "NamedNumberList" shall be different.

14.7 The order of the "NamedNumber" sequences in the "NamedNumberList" is not significant.

14.8 The tag for types defined by this notation is universal class, number 2.

14.9 The value of an integer type shall be defined by the notation "IntegerValue":

IntegerValue ::=

SignedNumber |
identifier

14.10 The "identifier" in "IntegerValue" shall be equal to that of an "identifier" in the "IntegerType" sequence with which the value is associated, and shall represent the corresponding number.

Note – When defining an integer value for which an "identifier" has been defined, use of the "identifier" form of "IntegerValue" should be preferred.

15 Notation for the enumerated type

15.1 The enumerated type (see § 3.17) shall be referenced by the notation "EnumeratedType":

EnumeratedType ::= ENUMERATED {Enumeration}

Enumeration ::=

NamedNumber |
NamedNumber, Enumeration

Note 1 – Each value has an identifier which is associated, in this notation, with a distinct integer. This provides control of the representation of the value in order to facilitate compatible extensions, but the values themselves are not expected to have any integer semantics.

Note 2 – The numeric values inside the "NamedNumber" in the "Enumeration" are not necessarily ordered or contiguous.

15.2 For each "NamedNumber", the "identifier" and the "SignedNumber" shall be distinct from all other "identifier"s and "SignedNumber"s in the "Enumeration".

15.3 The enumerated type has a tag which is universal class, number 10.

15.4 The value of an enumerated type shall be defined by the notation "EnumeratedValue":

EnumeratedValue ::= identifier

16 Notation for the real type

16.1 The real type (see § 3.18) shall be referenced by the notation "RealType":

RealType ::= REAL

16.2 The values of the real type are the values PLUS-INFINITY and MINUS-INFINITY together with the real numbers capable of being specified by the following formula involving three integers, M, B and E:

$$M \times B^E$$

where M is called the mantissa, B the base, and E the exponent. M and E may take any integer values, positive or negative, while B can take the values 2 or 10. All combinations of M, B and E are permitted.

Note 1 – This type is capable of carrying an exact representation of any number which can be stored in typical floating point hardware, and of any number with a finite character decimal representation.

Note 2 – The encoding (of this type) which is specified in Recommendation X.209 allows use of base 2, 8 or 16 with a binary representation of real values, and base 10 with a character representation. The choice is a sender's option.

16.3 The real type has a tag which is universal class, number 9.

16.4 The notation for defining a value of a real type shall be "RealValue":

RealValue ::= NumericRealValue | SpecialRealValue

NumericRealValue ::=

{Mantissa, Base, Exponent} | 0

Mantissa ::= SignedNumber

Base ::= 2 | 10

Exponent ::= SignedNumber

SpecialRealValue ::= PLUS-INFINITY | MINUS-INFINITY

The form "0" shall be used for zero values, and the alternate form for "NumericRealValue" shall not be used for zero values.

17 Notation for the bitstring type

17.1 The bitstring type (see § 3.19) shall be referenced by the notation "BitStringType":

BitStringType ::=

BIT STRING |
BIT STRING{NamedBitList}

NamedBitList ::=

NamedBit |
NamedBitList,NamedBit

NamedBit ::=

identifier(number) |
identifier(DefinedValue)

17.2 The "NamedBitList" is not significant in the definition of a type. It is used solely in the value notation specified in § 17.8.

17.3 The first bit in a bitstring has the number **zero**. The final bit in a bit string is called the **trailing bit**.

Note – This terminology is used in specifying the value notation and the encoding rules.

17.4 The "DefinedValue" shall be a reference to a non-negative value of type integer or enumerated, or of a type derived from those by tagging or subtyping.

17.5 The value of each "number" or "DefinedValue" appearing in the "NamedBitList" shall be different, and is the number of a distinguished bit in a bitstring value.

17.6 Each "identifier" appearing in the "NamedBitList" shall be different.

Note – The order of the "NamedBit" sequences in the "NamedBitList" is not significant.

17.7 This type has a tag which is universal class, number 3.

17.8 The value of a bitstring type shall be defined by the notation "BitStringValue":

BitStringValue ::=

bstring |
hstring |
{IdentifierList} |
{}

IdentifierList ::=

identifier |
IdentifierList,identifier

17.9 Each "identifier" in "BitStringValue" shall be the same as an "identifier" in the "BitStringType" sequence with which the value is associated.

17.10 The use of the notation determines, and can indicate by comment, whether or not the presence or absence of trailing zero bits is significant.

Note – Encoding rules enable the transfer of an arbitrary pattern, arbitrary length, string of bits.

17.11 The "{IdentifierList}" and "{ }" notations for "BitStringValue" shall not be used if the presence or absence of trailing zero bits is significant. This notation denotes a bitstring value with ones in the bit positions specified by the numbers corresponding to the "identifier" sequences, and with all other bits zero.

Note – The "{ }" sequence is used to denote a bitstring value which contains no one bits.

17.12 In specifying the encoding rules for a bitstring, the bits shall be referenced by the terms **first bit** and **trailing bit**, as defined above.

17.13 When using the "bstring" notation, the **first bit** is on the left, and the **trailing bit** is on the right.

17.14 When using the "hstring" notation, the most significant bit of each hexadecimal digit corresponds to the earlier (leftmost) bit in the bitstring.

Note – This notation does not in any way constrain the way encoding rules place a bitstring into octets for transfer.

17.15 The "hstring" notation shall not be used unless either:

a) the bitstring value consists of a multiple of four bits; or

b) the presence or absence of trailing zero bits is not significant.

Example:

'A98A'H

and

'1010100110001010'B

are alternative notations for the same bitstring value.

18 Notation for the octetstring type

18.1 The octetstring type (see § 3.20) shall be referenced by the notation "OctetStringType":

OctetStringType ::= OCTET STRING

18.2 This type has a tag which is universal class, number 4.

18.3 The value of an octetstring type shall be defined by the notation "OctetStringValue":

OctetStringValue ::=

bstring |
hstring

18.4 In specifying the encoding rules for an octetstring, the octets are referenced by the terms **first octet** and **trailing octet**, and the bits within an octet are referenced by the terms **most significant** and **least significant bit**.

18.5 When using the "bstring" notation, the left-most bit shall be the most significant bit of the first octet. If the "bstring" is not a multiple of eight bits, it shall be interpreted as if it contained additional zero trailing bits to make it the next multiple of eight.

18.6 When using the "hstring" notation, the left-most hexadecimal digit shall be the most significant semi-octet of the first octet. If the "hstring" is not an even number of hexadecimal digits, it shall be interpreted as if it contained a single additional trailing zero hexadecimal digit.

19 Notation for the null type

19.1 The null type (see § 3.21) shall be referenced by the notation "NullType":

NullType ::= NULL

19.2 This type has a tag which is universal class, number 5.

19.3 The value of the null type shall be referenced by the notation "NullValue":

NullValue ::= NULL

20 Notation for sequence types

20.1 The notation for defining a sequence type (see § 3.22) from other types shall be the "SequenceType":

SequenceType ::=

SEQUENCE{ElementTypeList} |
SEQUENCE{ }

ElementTypeList ::=

ElementType |
ElementTypeList,ElementType

ElementType ::=

NamedType |
NamedType OPTIONAL |
NamedType DEFAULT Value |
COMPONENTS OF Type

20.2 The "Type" in the fourth alternative of the "ElementType" shall be a sequence type. The "COMPONENTS OF Type" notation shall be used to define the inclusion, at this point in the "ElementTypeList", of all the "ElementType" sequences appearing in the referenced type.

Note – This transformation is logically completed prior to the satisfaction of the requirements in the following clauses.

20.3 For each series of one or more consecutive "ElementTypes" marked as OPTIONAL or DEFAULT, the tags of those "ElementTypes" and of any immediately following "ElementType" shall be distinct (see § 26).

20.4 If "OPTIONAL" or "DEFAULT" are present, the corresponding value may be omitted from a value of the new type, and from the information transferred by encoding rules.

Note 1 – The value notation may be ambiguous in this case, unless "identifier" sequences are present in each NamedType.

Note 2 – Encoding rules ensure that the encoding for a sequence value in which a "DEFAULT" or "OPTIONAL" element value is omitted is the same as that for a sequence value of a type in whose type definition the corresponding element was omitted. This feature can be useful in defining subsets.

20.5 If "DEFAULT" occurs, the omission of a value for that type shall be exactly equivalent to the insertion of the value defined by "Value", which shall be a value signification that is valid for the type defined by "Type" in the "NamedType" sequence.

20.6 The "identifier" (if any) in all "NamedType" sequences of the "ElementTypeList" shall be distinct.

20.7 All sequence types have a tag which is universal class, number 16.

Note – Sequence-of types have the same tag (see § 21.3).

20.8 The notation for defining the value of a sequence type shall be "SequenceValue":

SequenceValue ::=

{ElementValueList} |
{ }

ElementValueList ::=

NamedValue |
ElementValueList,NamedValue

20.9 The "{ }" notation shall only be used if:

a) all "ElementType" sequences in the "SequenceType" are marked "DEFAULT" or "OPTIONAL", and all values are omitted; or

b) the type notation was "SEQUENCE{ }".

20.10 There shall be one "NamedValue" for each "NamedType" in the "SequenceType" which is not marked OPTIONAL or DEFAULT, and the values shall be in the same order as the corresponding "NamedType" sequences.

Note – The use of "NamedType" sequences which do not contain an identifier is not prohibited, but can render the value notation ambiguous if "OPTIONAL" or "DEFAULT" is used.

21 Notation for sequence-of types

21.1 The notation for defining a sequence-of type (see § 3.23) from another type shall be the "SequenceOf-Type".

SequenceOfType ::=

SEQUENCE OF Type |
SEQUENCE

21.2 The notation "SEQUENCE" is synonymous with the notation "SEQUENCE OF ANY" (see § 27).

21.3 All sequence-of types have a tag which is universal class, number 16.

Note – Sequence types have the same tag (see § 20.7).

21.4 The notation for defining a value of a sequence-of type shall be the "SequenceOfValue":

SequenceOfValue ::= {ValueList} | { }

ValueList ::=

Value |
ValueList,Value

The "{ }" notation is used when there are no component values in the sequence-of value.

21.5 Each "Value" sequence in the "ValueList" shall be the notation for a value of the "Type" specified in the "SequenceofType".

Note – Semantic significance may be placed on the order of these values.

22 Notation for set types

22.1 The notation for defining a set type (see § 3.24) from other types shall be the "SetType".

SetType ::=

SET{ElementTypeList} |
SET{ }

"ElementTypeList" is specified in § 20.1

22.2 The "Type" in the fourth alternative of the "ElementType" (see § 20.1) shall be a set type. The "COMPONENTS OF Type" notation shall be used to define the inclusion of all the "ElementType" sequences appearing in the referenced type.

Note – This transformation is logically completed prior to the satisfaction of the requirements in the following clauses.

22.3 The "ElementType" types in a set type shall all have different tags. (See § 26.)

22.4 Sub-clauses § 20.4, § 20.5 and § 20.6 also apply to set types.

22.5 All set types have a tag which is universal class, number 17.

Note – Set-of types have the same tag (see § 23.3).

22.6 There shall be no more semantic associated with the order of values in a set type.

22.7 The notation for defining the value of a set type shall be "SetValue":

SetValue ::= {ElementValueList} | {}

"ElementValueList" is specified in § 20.8.

22.8 The "SetValue" shall only be "{}" if:

a) all "ElementType" sequences in the "SetType" are marked "DEFAULT" or "OPTIONAL", and all values are omitted; or

b) the type notation was "SET"{}".

22.9 There shall be one "NamedValue" for each "NamedType" in the "SetType" which is not marked "OPTIONAL" or "DEFAULT".

Note 1 – These "NamedValues" may appear in any order.

Note 2 – The use of "NamedType" sequences which do not contain an identifier is not prohibited, but can render the value notation ambiguous.

23 Notation for set-of types

23.1 The notation for defining a set-of type (see § 3.25) from another type shall be the "Set OfType":

SetOfType ::= SET OF Type |
SET

23.2 The notation "SET" is synonymous with the notation "SET OF ANY" (see § 27).

23.3 All set-of types have a tag which is universal class, number 17.

Note – Set types have the same tag (see § 22.5).

23.4 The notation for defining a value of a set-of type shall be the "SetOfValue":

SetOfValue ::= {ValueList} | {}

"ValueList" is specified in § 21.4

The "{}" notation is used when there are no component values in the set-of values.

23.5 Each "Value" sequence in the "ValueList" shall be the notation for a value of the "Type" specified in the "SetofType".

Note 1 – Semantic significance should not be placed on the order of these values.

Note 2 – Encoding rules are not required to preserve the order of these values.

24 Notation for choice types

24.1 The notation for defining a choice type (see § 3.27) from other types shall be the "ChoiceType":

ChoiceType ::= CHOICE{AlternativeTypeList}

AlternativeTypeList ::=

NamedType |
AlternativeTypeList,NamedType

Note 1 – The encoding rules encode the chosen alternative in a way which is indistinguishable from a "Type" consisting only of the "Type" contained in that alternative.

Note 2 – Specifying a "ChoiceType" with a single "NamedType" in the "AlternativeTypeList" cannot be distinguished in any encoding of a value from direct use of the "Type" in the "NamedType".

24.2 The types defined in the "AlternativeTypeList" shall all have distinct tags. (See § 26.)

24.3 The tag of the choice type shall be considered to be variable. When a value is selected, the tag becomes equal to the tag of the "Type" in the "NamedType" in the "AlternativeTypeList" from which the value is taken.

24.4 Where this type is used in a place where this Recommendation requires the use of types with distinct tags (see § 20.3, § 22.3 and § 24.2), the tags of all types defined in the "AlternativeTypeList" shall differ from those of the other types. (See § 26.) The following examples illustrate this requirement. Examples 1 and 2 are correct uses of the notation. Example 3 is incorrect, as the tags for types d and f, and e and g are identical.

Example 1:

```
A ::=  CHOICE
       {b B,
        c NULL}

B ::=  CHOICE
       {d [0] NULL,
        e [1] NULL}
```

Example 2:

```
A ::=  CHOICE
       {b B,
        c C}

B ::=  CHOICE
       {d [0] NULL,
        e [1] NULL}

C ::=  CHOICE
       {f [2] NULL,
        g [3] NULL}
```

Example 3:

```
(INCORRECT)
A ::=  CHOICE
       {b B,
        c C}

B ::=  CHOICE
       {d [0] NULL,
        e [1] NULL}

C ::=  CHOICE
       {f [0] NULL,
        g [1] NULL}
```

24.5 The "identifier" (if any) in all "NamedType" sequences of the "AlternativeTypeList" shall be distinct.

24.6 Where this type is used in a place where this Recommendation requires the use of "NamedTypes" with distinct "identifiers", the "identifiers" (if any) of all "NamedTypes" in the "AlternativeTypeList" shall differ from those (if any) of the other "NamedTypes".

24.7 The notation for defining the value of a choice type shall be the "ChoiceValue":

ChoiceValue ::= NamedValue

24.8 If the "NamedValue" contains an "identifier", it shall be a notation for a value of that type in the "AlternativeTypeList" that is named by the same "identifier". If the "NamedValue" does not contain an "identifier", it shall be a notation for a value of one of those types in the "AlternativeTypeList" that are not named by an "identifier".

Note – Failure to use an "identifier" in the "NamedType" can make the value notation ambiguous.

25 Notation for selection types

25.1 A "NamedType" appearing in the "AlternativeTypeList" of a "ChoiceType" can be referenced by the notation "SelectionType":

SelectionType ::= identifier < Type

where "Type" is a notation referencing the "ChoiceType", and "identifier" is the "identifier" in the "NamedType".

Note – "SelectionType" can be used either as a "NamedType", in which case the "identifier" is used in the value notation, or as a "Type" within a "NamedType", in which case its "identifier" is not used.

25.2 The notation for a value of a selection type shall be "SelectionValue":

SelectionValue ::= NamedValue

where the "NamedValue" contains the identifier that appears in the corresponding "SelectionType" if the "SelectionType" is used as a "NamedType", but not otherwise.

26 Notation for tagged types

A tagged type (see § 3.26) is a new type which is isomorphic with an old type, but which has a different tag. In all encoding schemes, a value of the new type can be distinguished from a value of the old type. The tagged type is mainly of use where this Recommendation requires the use of types with distinct tags. (See § 20.3, § 22.3, § 24.2, § 24.4, and § 27.6.)

Note – Where a protocol determines that values from several datatypes may be transmitted at any moment in time, distinct tags may be needed to enable the recipient to correctly decode the value.

26.1 The notation for a tagged type shall be "TaggedType":

```
TaggedType ::=

        Tag Type |
        Tag IMPLICIT Type |
        Tag EXPLICIT Type

Tag ::= [Class ClassNumber]

ClassNumber ::=

        number |
        DefinedValue

Class ::=

        UNIVERSAL |
        APPLICATION |
        PRIVATE |
        empty
```

26.2 The "DefinedValue" shall be a reference to a non-negative value of type integer, or of a type derived from type integer by tagging.

26.3 The new type is isomorphic with the old type, but has a tag with "Class" class and number "Class-Number", unless the "Class" is "empty", when the tag is context-specific class, number "ClassNumber".

26.4 The "Class" shall not be "UNIVERSAL" except for types defined in this Recommendation.

Note – Use of universal class tags is agreed from time to time by ISO and CCITT.

26.5 If the "Class" is "APPLICATION", the same "Tag" shall not be used again in the same module.

26.6 If the "Class" is "PRIVATE" the "Tag" is available for use on an enterprise-specific basis.

26.7 The tagging construction specifies explicit tagging if any of the following holds:

a) the "Tag EXPLICIT Type" alternative is used;

b) the "Tag Type" alternative is used and the value of "TagDefault" for the module is "EXPLICIT TAGS";

c) the "Tag Type" alternative is used and the value of "TagDefault" for the module is "IMPLICIT TAGS", but the type defined by "Type" is a choice type or on any type.

The tagging construction specifies implicit tagging otherwise.

26.8 If the "Class" is "empty", there are no restrictions on the use of "Tag", other than those implied by the requirement for distinct tags in § 20.3, § 22.3, and § 24.2.

26.9 Implicit tagging indicates, for those encoding rules which provide the option, that explicit identification of the tag of the "Type" in the "TaggedType" is not needed during transfer.

Note – It can be useful to retain the old tag where this was universal class, and hence unambiguously identifies the old type without knowledge of the ASN.1 definition of the new type. Minimum transfer octets is, however, normally achieved by the use of IMPLICIT. An example of encoding using IMPLICIT is given in Recommendation X.209.

26.10 The "IMPLICIT" alternative shall not be used if the type defined by "Type" is a choice type or an any type.

26.11 The notation for a value of a "TaggedType" shall be "TaggedValue":

TaggedValue ::= Value

where "value" is the notation for a value of the "Type" in the "TaggedType".

Note – The "Tag" does not appear in this notation.

27 Notation for the any type

27.1 The notation for any type (see § 3.29) is "AnyType":

```
AnyType ::=

        ANY |
        ANY DEFINED BY identifier
```

Note – The use of "ANY" in an ISO Standard or CCITT Recommendation produces an incomplete specification unless it is supplemented by additional specification. The "ANY DEFINED BY" construct provides the means of specifying in an instance of communication the type which fills the ANY, and a pointer to its semantics. If the following rules for its use are followed, it can provide a complete specification. Use of ANY without the DEFINED BY construct is deprecated.

27.2 The "DEFINED BY" alternative shall be used only when the any type, or a type derived from it by tagging, is one of the component types of a sequence type or set type (the containing type).

27.3 The "identifier" in the "DEFINED BY" alternative shall also appear in a "NamedType" that specifies another, non-optional, component of the containing type. The "NamedType" shall be an integer type or an enumerated type or an object identifier type or a type derived from those by tagging or subtyping.

27.4 When the "NamedType" is an integer or enumerated type, or of a type derived from those types by tagging ot subtyping the document employing the "DEFINED BY" notation shall contain, or explicitly reference, a single list which specifies the ASN.1 type to be carried by the ANY for each permitted value of the integer type. There shall be precisely any one such list in all instances of communication of the containing type.

27.5 When the "NamedType" is an object identifier type, or a type derived from object identifier by tagging, there is a need for registers which, for each allocated object identifier value, associate a single ASN.1 type (which may be a CHOICE type) which is to be carried by the ANY.

Note 1 – There may be an arbitrary number of registers associating an object identifier value with an ASN.1 type for this purpose.

Note 2 – Registration of values for open interconnection is expected to occur within ISO Standards and CCITT Recommendations using the notation. Where a separate International Registration Authority is intended for any instance of "ANY DEFINED BY", this should be identified in the document using the notation.

Note 3 – The main difference between the integer and object identifier definers is that the use of integer references a single list, contained in the using standard or Recommendation, whilst the use of object identifier allows an open-ended set of types determined by any authority able to allocate object identifiers.

27.6 This type has an indeterminate tag, and shall not be used where this Recommendation requires distinct tags. (See § 20.3, § 22.3, § 24.2 and § 24.4.)

27.7 The notation for the value of an any type shall be defined using ASN.1 and is "AnyValue":

```
AnyValue ::= Type Value
```

where "Type" is the notation for the chosen type, and "Value" is the notation for a value of this type.

28 Notation for the object identifier type

28.1 The object identifier type (see § 3.34) shall be referenced by the notation "ObjectIdentifierType":

```
ObjectIdentifierType ::=
        OBJECT IDENTIFIER
```

28.2 This type has a tag which is universal class, number 6.

28.3 The value notation for an object identifier shall be "ObjectIdentifierValue":

```
ObjectIdentifierValue ::=
        {ObjIdComponentList} |
        {DefinedValue ObjIdComponentList}
ObjIdComponentList ::=
        ObjIdComponent |
        ObjIdComponent ObjIdComponentList
ObjIdComponent ::=
        NameForm |
        NumberForm |
        NameAndNumberForm
```

NameForm ::= identifier

NumberForm ::= number | DefinedValue

NameAndNumberForm ::=

identifier(NumberForm)

28.4 The "DefinedValue" in "NumberForm" shall be a reference to a value of type integer or enumerated, or of a type derived from those by tagging or subtyping.

28.5 The "DefinedValue" in "ObjectIdentifierValue" shall be a reference to a value of type object identifier, or of a type derived from object identifier by tagging.

28.6 The "NameForm" shall be used only for those object identifier components whose numeric value and identifier are specified in annexes B to D, and shall be one of the identifiers specified in annexes B to D.

28.7 The "number" in the "NumberForm" shall be the numeric value assigned to the object identifier component.

28.8 The "identifier" in the "NameAndNumberForm" shall be specified when a numeric value is assigned to the object identifier component.

Note – The authorities allocating numeric values to object identifier components are identified in the annexes to this Recommendation.

28.9 The semantics of an object identifier value are defined by reference to an **object identifier tree**. An object identifier tree is a tree whose root corresponds to this Recommendation and whose vertices correspond to administrative authorities responsible for allocating arcs from that vertex. Each arc of the tree is labelled by an object identifier component which is a numeric value. Each information object to be identified is allocated precisely one vertex (normally a leaf), and no other information object (of the same or a different type) is allocated to that same vertex. Thus an information object is uniquely and unambiguously identified by the sequence of numeric values (object identifier components) labelling the arcs in a path from the root to the vertex allocated to the information object.

Note – Object identifier values contain at least two object identifier components, as specified in annexes B to D.

28.10 An object identifier value is semantically an ordered list of object identifier component values. Starting with the root of the object identifier tree, each object identifier component value identifies an arc in the object identifier tree. The last object identifier component value identifies an arc leading to a vertex which an information object has been assigned. It is this information object which is identified by the object identifier value. The significant part of the object identifier component is the "NameForm" or "NumberForm" which it reduces to, and which provides the numeric value for the object identifier component.

Note – In general, an information object is a class of information (for example, a file format), rather than an instance of such a class (for example, an individual file). It is thus the class of information (defined by some referencable specification), rather than the piece of information itself, that is assigned a place in the tree.

28.11 Where the "ObjectIdentifierValue" includes a "DefinedValue", the list of object identifier components to which it refers is prefixed to the components explicitly present in the value.

Examples: With identifiers is assigned as specified in Annex B, the values:

{iso standard 8571 pci (1)}

and

{1 0 8571 1}

would each identify an object, "pci", defined in ISO 8571.

With the following additional definition:

ftam OBJECT IDENTIFIER ::=

{iso standard 8571}

the following value is also equivalent to those above:

{ftam pci (1)}

Note – It is recommended that, whenever a CCITT Recommendation, ISO standard or other document assigns values of type OBJECT IDENTIFIER to information objects there should be an appendix or annex which summarizes the assignments made therein. It is also recommended that an authority assigning values of type OBJECT IDENTIFIER to an information object should also assign values of type ObjectDescriptor to that information object.

29 Notation for character string types

29.1 The notation for referencing a character string type (see § 3.12 and section two) shall be:

CharacterStringType ::= typereference

where "typereference" is one of the character string type names listed in section two.

29.2 The tag of each character string type is specified in section two.

29.3 The notation for a character string value shall be:

CharacterStringValue ::= cstring

The definition of the character string type determines the characters appearing in the "cstring".

30 Notation for types defined in section three

30.1 The notation for referencing a type defined in section three of this Recommendation shall be:

UsefulType ::= typereference

where "typereference" is one of those defined in section three using the ASN.1 notation.

30.2 The tag of each "UsefulType" is specified in section three.

30.3 The notation for a value of a "UsefulType" is specified in section three.

SECTION 2 – CHARACTER STRING TYPES

31 Definition of character string types

This clause defines types whose distinguished values are sequences of zero, one or more characters from some character set.

31.1 The type is defined by specifying:

a) the tag assigned to the type; and

b) a name by which the type definition can be referenced; and

c) the characters in the character set used in defining the type, either by reference to a table listing the character graphics or by reference to a registration number in the ISO International Register of Coded Character Sets to be used with Escape Sequences.

The name in b) above may be used as a "typereference" in the ASN.1 notation (see § 29).

31.2 Table 6/X.208 lists the name by which each of these type definitions can be referenced, the number of the universal class tag assigned to the type, the defining registration numbers or following table, and, where necessary, identification of a NOTE relating to the entry in the table. Where a synonymous name is defined in the notation, this is listed in parentheses.

Note – The tag assigned to character string types unambiguously identifies the type. Note, however, that if ASN.1 is used to define new types from this type (particularly using IMPLICIT), it may be impossible to recognize these types without knowledge of the ASN.1 type definition.

31.3 Table 4/X.208 lists the characters which can appear in the NumericString type.

TABLE 4/X.208

NumericString

Name	Graphic
Digits	0, 1, ... 9
Space	(space)

TABLE 5/X.208

PrintableString

Name	Graphic
Capital letters	A, B, ... Z
Small letters	a, b, ... z
Digits	0, 1, ... 9
Space	(space)
Apostrophe	'
Left Parenthesis	(
Right Parenthesis	)
Plus sign	+
Comma	,
Hyphen	-
Full stop	.
Solidus	/
Colon	:
Equal sign	=
Question mark	?

TABLE 6/X.208

List of character string types

Name for referencing the type	Universal class number	Defining registration numbers (see ISO 2375) or table number	Notes
NumericString	18	Table 4	1
PrintableString	19	Table 5	1
TeletexString (T61String)	20	87, 102, 103, 106, 107 + SPACE + DELETE	2
VideotexString	21	1, 72, 73, 102, 108, 128, 129 + SPACE + DELETE	3
VisibleString (ISO646String)	26	2 + SPACE	
IA5String	22	1, 2 + SPACE + DELETE	
GraphicString	25	All G sets + SPACE	
GeneralString	27	All G and all C sets + SPACE + DELETE	

Note 1 – The type-style, size, colour, intensity, or other display characteristics are not significant.

Note 2 – The entries corresponding to these registration numbers reference CCITT Recommendation T.61 for rules concerning their use.

Note 3 – The entries corresponding to these registration numbers provide the functionality of CCITT Recommendations T.100 and T.101.

31.4 Table 5/X.208 lists the characters which can appear in the PrintableString type.

31.5 The notation for these types shall be "cstring".

Note – This notation can only be used on a medium capable of displaying the characters which are present in the value. The notation for the value in other cases is not defined.

31.6 In all cases, the range of permitted characters may be restricted by a comment, but shall not be extended.

SECTION 3 – USEFUL DEFINITIONS

This section contains definitions which are expected to be useful in a number of applications.

Note – It is expected that this section will be added to, to encompass other common datatypes such as diagnostics, authentication information, accounting information, security parameters and so on.

The value notation and semantic definition for types defined in this section are derived from a definition of the type using the ASN.1 notation. This type definition can be referenced by standards or Recommendations defining encoding rules in order to specify encodings for these types.

32 Generalized time

32.1 This type shall be referenced by the name:

GeneralizedTime

32.2 The type consists of values representing:

a) a calendar date, as defined in ISO 2014 (Writing of calendar dates in all-numeric form); and

b) a time of day, to any of the precisions defined in clause 2 of ISO 3307 (Representations of time of day); and

c) the local differential factor as defined in ISO 4031 (Representation of local time differentials).

32.3 The type can be defined, using ASN.1, as follows:

GeneralizedTime ::=

[UNIVERSAL 24] IMPLICIT
VisibleString

with the values of the "VisibleString" restricted to strings of characters which are either:

a) a string representing the calendar date, as specified in ISO 2014, with a four-digit representation of the year, a two-digit representation of the month and a two-digit representation of the day, without use of separators, followed by a string representing the time of day, as specified in ISO 3307, without separators other than decimal comma or decimal period (as provided for in clauses 2.3, 2.4 and 2.5 of ISO 3307), and with no terminating Z (as provided for in clause 3 of ISO 3307); or

b) the characters in a) above followed by an upper-case letter Z; or

c) the characters in a) above followed by a string representing a local time differential, as specified in ISO 4031, without separators.

In case a), the time shall represent the local time. In case b), the time shall represent UTC time. In case c), the part of the string formed in case a) represents the local time (t_1), and the time differential (t_2) enables UTC time to be determined as follows:

UTC time is $t_1 - t_2$

Examples:

Case a)	19851106210627.3 local time 6 minutes, 27.3 seconds after 9 pm on 6 November 1985.
Case b)	19851106210627.3Z UTC time as above.
Case c)	19851106210627.3 – 0500 Local time as in example a), with local time 5 hours retarded in relation to UTC time.

32.4 The tag shall be as defined in § 32.3.

32.5 The value notation shall be the value notation for the "VisibleString" defined in § 32.3.

33 Universal time

33.1 This type shall be referenced by the name:

UTCTime

33.2 The type consists of values representing:

a) a calendar date; and

b) a time to a precision of one minute or one second; and

c) (optionally) a local time differential from coordinated universal time.

33.3 The type can be defined, using ASN.1, as follows:

UTCTime ::=

[UNIVERSAL 23]IMPLICIT
VisibleString

with the values of the "VisibleString" restricted to strings of characters which are the juxtaposition of:

a) the six digits YYMMDD where YY is the two low-order digits of the Christian year, MM is the month (counting January as 01), and DD is the day of the month (01 to 31); and

b) either:

1) the four digits hhmm where hh is hour (00 to 23) and mm is minutes (00 to 59); or

2) the six digits hhmmss where hh and mm are as in 1) above, and ss is seconds (00 to 59); and

c) either:

1) the character Z; or

2) one of the characters + or −, followed by hhmm, where hh is hour and mm is minutes.

The alternatives in b) above allow varying precisions in the specification of the time.

In alternative c) 1), the time is UTC time. In alternative c) 2), the time (t_1) specified by a) and b) above is the local time; the time differential (t_2) specified by c) 2) above enables the UTC time to be determined as follows:

UTC Time is $t_1 - t_2$

Example: If local time is 7 AM on 2 January and coordinated universal time is 12 noon on 2 January, the value is either of:

UTCTime "8201021200Z"
UTCTime "8201020700−0500"

33.4 The tag shall be as defined in § 33.3.

33.5 The value notation shall be the value notation for the "VisibleString" defined in § 33.3.

34 The external type

34.1 The notation for an external type (see § 3.30) is "ExternalType":

ExternalType ::= EXTERNAL

34.2 The type consists of values representing:

a) an encoding of a single data value that may, but need not, be the value of a single ASN.1 datatype; and

b) identification information which determines the semantics and encoding rules; and

c) (optionally) an object descriptor which describes the object.

The optional object descriptor shall not be present unless explicitly permitted by comment associated with the use of the EXTERNAL notation.

34.3 Type EXTERNAL permits the inclusion of any data value from an identified set of data values.

Note 1 – The specification of this set of data values, their semantics, the assignment of an object identifier and (optionally) an object descriptor, and the dissemination of this information to all communicating parties is called **registering an abstract syntax**. This operation can be performed by any authority entitled to allocate an OBJECT IDENTIFIER value, as specified in Annexes B to D.

Note 2 – A set of data values registered as an abstract syntax (with associated encoding rules) is not well-formed unless the encoding of each data value is self-identifying within the set of data value encodings. When ASN.1 is used to define an abstract syntax, tagging is used to provide self-identification. Where an abstract syntax is not well-formed, use of the communications channel is either context-sensitive or leads to ambiguity.

34.4 The EXTERNAL type can be defined, using ASN.1, as follows:

```
EXTERNAL ::= [UNIVERSAL 8] IMPLICIT SEQUENCE

{direct-reference          OBJECT IDENTIFIER OPTIONAL,
indirect-reference         INTEGER OPTIONAL,
data-value-descriptor      ObjectDescriptor OPTIONAL,
encoding                   CHOICE
  {single-ASN1-type        [0] ANY,
   octet-aligned           [1] IMPLICIT OCTET STRING,
   arbitrary               [2] IMPLICIT BIT STRING}}
```

34.5 When presentation layer negotiation of encoding rules is not in use (prior agreement of transfer syntax) for the value of this EXTERNAL, the "direct-reference OBJECT IDENTIFIER" shall be present. In this case the identifier of the set of data values is an object identifier which directly references an abstract syntax and fills the "direct-reference OBJECT IDENTIFIER" field of the "EXTERNAL". In this case, the abstract syntax registration also defines the encoding rules (transfer syntax) for the data value and the "indirect-reference INTEGER" shall not be included.

34.6 When presentation layer negotiation is in use for the value of this EXTERNAL, the "indirect-reference INTEGER" shall be present. In this case the identifier of the set of data values is an integer which references an instance of use of an abstract syntax. The integer is called a **presentation context identifier** and fills the "indirect-reference INTEGER" field of the "EXTERNAL". If presentation layer negotiation has been completed, the presentation context identifier also identifies the encoding rules (transfer syntax) for the data value and the "direct-reference OBJECT IDENTIFIER" shall not be included. If presentation layer negotiation is not complete, an object identifier value is also needed which identifies the encoding rules (transfer syntax) used for the encoding. Where presentation layer negotiation is in use, and where the "direct-reference OBJECT IDENTIFIER" element is allowed or required to carry such a value, this shall be identified by comment associated with the use of the "EXTERNAL" notation, otherwise the field shall be absent.

Note 1 – The effect of § 34.5 and § 34.6 is to make the presence of at least one of the "direct-reference" and the "indirect-reference" mandatory.

Note 2 – Both references are present when presentation layer negotiation is in use but incomplete.

34.7 If the data value is the value of a single ASN.1 datatype, and if the encoding rules for this data value are the same as those for the complete "EXTERNAL" datatype, then the sending implementation shall use any of the "Encoding" choices:

single-ASN1-type
octet-aligned
arbitrary

as an implementation option.

34.8 If the encoding of the data value, using the agreed or negotiated encoding, is an integral number of octets, then the sending implementation shall use any of the "Encoding" choices:

octet-aligned
arbitrary

as an implementation option.

Note – A data value which is a series of ASN.1 types, and for which the transfer syntax specifies simple concatenation of the octet strings produced by applying the ASN.1 Basic Encoding Rules to each ASN.1 type, falls into this category, not that of § 34.7.

34.9 If the encoding of the data value, using the agreed or negotiated encoding, is not an integral number of octets, the "Encoding" choice shall be:

arbitrary

34.10 If the "Encoding" choice is chosen as "single-ASN1-type", then the ASN.1 type shall replace the "ANY", with a value equal to the data value to be encoded.

Note – The range of values which might occur in the "ANY" is determined by the registration of the object identifier value associated with the "direct-reference", and/or the integer value associated with the "indirect-reference".

34.11 If the "Encoding" choice is chosen as "octet-aligned", then the data value shall be encoded according to the agreed or negotiated transfer syntax, and the result shall form the value of the bitstring.

34.12 If the "Encoding" choice is chosen as "arbitrary", then the data value shall be encoded according to the agreed or negotiated transfer syntax, and the result shall form the value of the bitstring.

34.13 The tag shall be as defined in § 34.4.

34.14 The value notation shall be the value of the type defined in § 34.4.

35 The object descriptor type

35.1 This type shall be referenced by the name:

ObjectDescriptor

35.2 The type consists of human-readable text which serves to describe an information object. The text is not an unambiguous identification of the information object, but identical text for different information objects is intended to be uncommon.

Note – It is recommended that an authority assigning values of type "OBJECT IDENTIFIER" to an information object should also assign values of type "ObjectDescriptor" to that information object.

35.3 The type can be defined, using the ASN.1 notation, as follows:

```
ObjectDescriptor ::=
        [UNIVERSAL 7] IMPLICIT
        GraphicString
```

The "GraphicString" contains the text describing the information object.

35.4 The tag shall be as defined in § 35.3.

35.5 The value notation shall be the value notation for the "GraphicString" defined in § 37.3.

SECTION 4 – SUBTYPES

36 Subtype notation

36.1 A subtype is defined by the notation for á parent type followed by an appropriate subtype specification. The subtype specification notation is made up of subtype value sets. The values in the subtype are determined as specified in § 36.7 by taking the union of all the subtype value sets.

36.2 The subtype notation shall not be used so as to produce a subtype with no values.

36.3 The notation for a subtype shall be "Subtype":

```
Subtype ::=
        ParentType SubtypeSpec |
        SET SizeConstraint OF Type |
        SEQUENCE SizeConstraint OF Type

ParentType ::= Type
```

36.4 When the "SubtypeSpec" notation follows the "SelectionType" notation, the parent type is the "SelectionType", not the "Type" in the "SelectionType" notation.

36.5 When the "SubtypeSpec" notation follows a set-of or sequence-of type notation, it applies to the "Type" in the set-of or sequence-of notation, not to the set-of or sequence-of type.

Note – The special notation "SET SizeConstraint OF" and "SEQUENCE Size Constraint OF" is used to provide an alternative mechanism (which is more readable than the general case notation) for simple cases. More complex cases require the general mechanism.

36.6 The subtype specification notation shall be "SubtypeSpec":

SubtypeSpec ::=

(SubtypeValueSet SubtypeValueSetList)

SubtypeList ::=

" | "
SubtypeValueSet
SubtypeValueSetList |
empty

36.7 Each "SubtypeValueSet" specifies a number (possibly zero) of values of the parent type, which are then included in the subtype. A value of the parent type is a value of the subtype if and only if it is included by one or more of the subtype value sets. The subtype is thus formed from the set union of the values included by the subtype value sets.

36.8 A number of different forms of notation for "SubtypeValueSet" are provided. They are identified below, and their syntax and semantics is defined in § 37. As specified in § 37, and summarized in Table 7/X.208, some notations can only be applied to particular parent types.

TABLE 7/X.208

Applicability of subtype value sets

Type (or derived from such a type by tagging)	Single Value	Cont'd Subtype	Value Range	Size Range	Alphabet Limitation	Inner Subtyping
Boolean	/	/	x	x	x	x
Integer	/	/	/	x	x	x
Enumerated	/	/	x	x	x	x
Real	/	/	/	x	x	x
Object Identifier	/	/	x	x	x	x
Bit String	/	/	x	/	x	x
Octet String	/	/	x	/	x	x
Character String Types	/	/	x	/	/	x
Sequence	/	/	x	x	x	/
Sequence-of	/	/	x	/	x	/
Set	/	/	x	x	x	/
Set-of	/	/	x	/	x	/
Any	/	/	x	x	x	x
Choice	/	/	x	x	x	/

Subtype ValueSet ::=

SingleValue |
ContainedSubtype |
ValueRange |
PermittedAlphabet | SizeConstraint | InnerTypeConstraints

37 Subtype Value Sets

37.1 *Single Value*

37.1.1 The "SingleValue" notation shall be:

SingleValue ::= Value

where "Value" is the value notation for the parent type.

37.1.2 A "SingleValue" set is the single value of the parent type specified by "Value". This notation can be applied to all parent types.

37.2 *Contained Subtype*

37.2.1 The "ContainedSubtype" notation shall be:

ContainedSubtype ::= INCLUDES Type

37.2.2 A "ContainedSubType" value consists of all the values of the "Type", which is itself required to be a subtype of the parent type. This notation can be applied to all parent types.

37.3 *Value Range*

37.3.1 The "ValueRange" notation shall be:

ValueRange ::= LowerEndpoint .. Upper Endpoint

37.3.2 A "ValueRange" value set consists of all the values in a range which are designated by specifying the numerical values of the endpoints of the range. This notation can only be applied to integer types, real types and types derived from those types by tagging or subtyping.

Note – For the purpose of subtyping, "PLUS-INFINITY" exceeds all "NumericReal" values and "MINUS-INFINITY" is less than all "NumericReal" values.

37.3.3 Each endpoint of the range is either closed (in which case that endpoint is included in the value set) or open (in which case the endpoint is not included). When open, the specification of the endpoint includes a less-than symbol ("<"):

LowerEndpoint ::= LowerEndValue | LowerEndValue <

UpperEndpoint ::= UpperEndValue | < UpperEndValue

37.3.4 An endpoint may also be unspecified, in which case the range extends in that direction as far as the parent type allows:

LowerEndValue ::= Value | MIN

UpperEndValue ::= Value | MAX

37.4 *Size Constraint*

37.4.1 The "SizeConstraint" notation shall be:

SizeConstraint ::= SIZE SubtypeSpec

37.4.2 A "SizeConstraint" can only be applied to bitstring types, octetstring types, character string types, set-of types or sequence-of types, or types formed from any of those types by tagging or subtyping.

37.4.3 The "SubtypeSpec" specifies the permitted integer values for the length of the numbers of the value set, and takes the form of any subtype specifications which can be applied to the following parent type:

INTEGER (0. .MAX)

37.4.4 The unit of measure depends on the parent type, as follows:

Type	**Unit of measure**
bit string	bit
octet string	octet
character string	character
set-of	component value
sequence-of	component value

37.5 *Permitted Alphabet*

37.5.1 The "PermittedAlphabet" notation shall be:

PermittedAlphabet ::= FROM SubtypeSpec

37.5.2 A "PermittedAlphabet" value consists of all values which can be constructed using a sub-alphabet of the parent string. This notation can only be applied to character string types, or to types formed from them by tagging or subtyping.

37.5.3 The "SubtypeSpec" specifies the characters which may appear in the character string, and is any subtype specification which can be applied to the subtype obtained by applying the subtype specification "SIZE(1)" to the parent type.

37.6 *Inner Subtyping*

37.6.1 The "InnerTypeConstraints" notation shall be:

InnerTypeConstraints ::=

WITH COMPONENT SingleTypeConstraint |
WITH COMPONENTS MultipleTypeConstraints

37.6.2 An "InnerTypeConstraints" includes in the value set only those values which satisfy a collection of constraints on the presence and/or values of the components of the parent type. A value of the parent type is not included in the subtype unless it satisfies all of the constraints expressed or implied (see § 37.6.6). This notation can be applied to the set-of, sequence-of, set, sequence and choice types, or types formed from them by tagging or subtyping.

37.6.3 For the types which are defined in terms of a single other (inner) type (set-of, sequence-of and types derived from them by tagging), a constraint taking the form of a subtype value specification is provided. The notation for this is "SingleTypeConstraint":

SingleTypeConstraint ::= SubtypeSpec

The "SubtypeSpec" defines a subtype of the single other (inner) type. A value of the parent type is a member of the subtype value set if and only if each inner value belongs to the subtype obtained by applying the "SubtypeSpec" to the inner type.

37.6.4 For the types which are defined in terms of multiple other (inner) types (choice, set, sequence, and types derived from them by tagging or subtyping), a number of constraints on these inner types can be provided. The notation for this is "MultipleTypeConstraints":

MultipleTypeConstraints ::=

FullSpecification | PartialSpecification

FullSpecification ::= {TypeConstraints}

PartialSpecification ::= {..., TypeConstraints}

TypeConstraints ::=

NamedConstraint |
NamedConstraint, TypeConstraints

NamedConstraint ::= identifier Constraint | Constraint

37.6.5 The "TypeConstraints" contains a list of constraints on the component types of the parent type. For a sequence type, the constraints must appear in order. The inner type to which the constraint applies is identified by means of its identifier, if it has one, or by its position, in the case of sequence types.

Note – Where the inner type has no identifier, the notation can be ambiguous.

37.6.6 The "MultipleTypeConstraints" comprises either a "FullSpecification" or a "PartialSpecification". Where "FullSpecification" is used, there is an implied presence constraint of "ABSENT" on all inner types not explicitly listed (see § 37.6.8), and each inner type which is not marked "OPTIONAL" or "DEFAULT" in the parent type shall be explicitly listed. Where "PartialSpecification" is employed, there are no implied constraints, and any inner type can be omitted from the list.

37.6.7 A particular inner type may be constrained in terms of its presence (in values of the parent type), its value, or both. The notation is "Constraint":

Constraint ::= ValueConstraint PresenceConstraint

37.6.8 A constraint on the value of an inner type is expressed by the notation "ValueConstraint":

ValueConstraint ::= SubtypeSpec | empty

The constraint is satisfied by a value of the parent type if and only if the inner value belongs to the subtype specified by the "SubtypeSpec" applied to the inner type.

37.6.9 A constraint on the presence of an inner type shall be expressed by the notation "PresenceConstraint":

PresenceConstraint ::= PRESENT | ABSENT | empty | OPTIONAL

The meaning of these alternatives, and the situations in which they are permitted, are defined in § 37.6.9.1 to § 37.6.9.3.

37.6.9.1 If the parent type is a sequence or set, an element type marked "OPTIONAL" may be constrained to be "PRESENT" (in which case the constraint is satisfied if and only if the corresponding element value is present) or to be "ABSENT" (in which case the constraint is satisfied if and only if the corresponding element value is absent or to be "OPTIONAL" (in which case the constraint is placed upon the presence of the corresponding element value).

37.6.9.2 If the parent type is a choice, a component type can be constrained to be "ABSENT", in which case the constraint is satisfied if and only if the corresponding component type is not used in the value.

37.6.9.3 The meaning of an empty "PresenceConstraint" depends on whether a "FullSpecification" or a "PartialSpecification" is being employed:

a) in a "FullSpecification", this is equivalent to a constraint of "PRESENT";

b) in a "PartialSpecification", no constraint is imposed.

ANNEX A

(to Recommendation X.208)

The macro notation

A.1 *Introduction*

A mechanism is provided within ASN.1 for the user of ASN.1 to define a new notation with which he can then construct and reference ASN.1 types or specify values of types. The new notation is defined using the notation "MacroDefinition". A "MacroDefinition" simultaneously specifies a new notation for constructing and referencing a type and also a new notation for specifying a value. (See § I.3 for an illustration of the use of the macro notation.)

With a "MacroDefinition" the ASN.1 user specifies the new notation by means of a set of productions in a manner similar to that of this Recommendation. The writer of the macro definition;

a) specifies the complete syntax to be used for defining all types supported by the macro (this syntax specification is invoked for syntax analysis by any occurrence of the macro name in the ASN.1 type notation); and

b) specifies the complete syntax to be used for a value of one of these types (this syntax specification is invoked for syntax analysis whenever a value of the macro type is expected); and

c) specifies, as the value of a standard ASN.1 type (of arbitrary complexity), the resulting type and value for all instances of the macro value notation.

An instance of the syntax defined by the macro definition can contain instances of types or values (using the standard ASN.1 notation). These types or values (appearing in the use of macro notation) can be associated, for the duration of the syntax analysis, with a **local type reference** or a **local value reference** by appropriate statements in the macro definition. It is also possible to embed, within the macro definition, standard ASN.1 type assignments. These assignments become active when the associated syntactic category is matched against an item or items in the instance of the new notation being analysed. Their lifetime is limited to that of the analysis.

When analysing a value in the new notation, assignments made during analysis of the corresponding type notation are available. Such analysis is considered to logically precede analysis of every instance of the value notation.

The resulting type and value of an instance of use of the new value notation is determined by the value (and the type of the value) finally assigned to the distinguished local value reference identified by the keyword item VALUE, according to the processing of the macrodefinition for the new type notation followed by that for the new value notation.

Each "MacroDefinition" defines a notation (a syntax) for type definition and a notation (a syntax) for value definition. The ASN.1 type which is defined by an instance of the new type notation may, but need not, depend on the instance of the value notation with which the type is associated. To this extent, the use of the new type notation is similar to a CHOICE – the tag is indeterminate. Thus the new notation cannot in this case be used in places where a known tag is required, nor can it be implicitly tagged.

A.2 *Extensions to the ASN.1 character set and items*

The characters | and > are used in the macro notation.

The items specified in the following subclauses are also used.

A.2.1 *Macroreference*

Name of item – macroreference

A «macroreference» shall consist of the sequence of characters specified for a "typereference" in § 8.2, except that all characters shall be in upper-case. Within a single module, the same sequence of characters shall not be used for both a typereference and a macroreference.

A.2.2 *Productionreference*

Name of item – productionreference

A «productionreference» shall consist of the sequence of characters specified for a "typereference" in § 8.2.

A.2.3 *Localtypereference*

Name of item – localtypereference

A "localtypereference" shall consist of the sequence of characters specified for a "typereference" in § 8.2. A "localtypereference" is used as an identifier for types which are recognized during syntax analysis of an instance of the new type or value notation.

A.2.4 *Localvaluereference*

Name of item – localvaluereference

A "localvaluereference" shall consist of the sequence of characters specified for a "valuereference" in § 8.2. A "localvaluereference" is used as an identifier for values which are recognized during syntax analysis of an instance of the new type or value notation.

A.2.5 *Alternation item*

Name of item – " | "

This item shall consist of the single character |.

A.2.6 *Definition terminator item*

Name of item – >

This item shall consist of the single character >.

Note – The item < for the start of definitions is defined in § 8.13.

A.2.7 *Syntactic terminal item*

Name of item – astring

An "astring" shall consist of an arbitrary number (possibly zero) of characters from the ASN.1 character set (see § 7), surrounded by ". The character " shall be represented in an "astring" by a pair of ".

Note – Use of "astring" in the macronotation specifies the occurrence, at the corresponding point in the syntax being analysed, of the characters enclosed in quotation marks (").

TABLE 8/X.208

Sequence specified by items

Item name	Defining clause
"string"	any sequence of characters
"identifier"	8.3 – Identifiers
"number"	8.8 – Numbers
"empty"	8.7 – Empty

A.2.8 *Syntactic category keywork items*

Names of items: "string"
"identifier"
"number"
"empty"

Items with the above names shall consist (in the macronotation) of the sequences of characters in the name, excluding the quotation symbols ("). These items are used in the macro notation to specify the occurrence, in an instance of the new notation, of certain sequences of characters. The sequences in the new notation specified by each item are given in Table 8/X.208 by reference to a clause in this Recommendation which defines the sequence of characters appearing in the new notation.

Note – The macro notation does not support the distinction between identifiers and references based on the case of the initial letter. This is for historical reasons.

A.2.9 *Additional keyword items*

Names of items: MACRO
TYPE
NOTATION
VALUE
value
type

Items with the above names shall consist of the sequence of characters in the name.

The items specified in clauses § A.2.2 to § A.2.4 inclusive shall not be one of the § A.2.9 sequences, except when used as specified below.

The keyword "MACRO" shall be used to introduce a macro definition. The keyword "TYPE NOTATION" shall be used as the name of the production which defines the new type notation. The keyword "VALUE NOTATION" shall be used as the name of the production which defines the new value notation. The keyword "VALUE" shall be used as the "localvaluereference" to which the resulting value is assigned. The keyword "value" shall be used to specify that each instance of the new notation contains at this point, using standard ASN.1 notation, some value of a type (specified in the macro definition). The keyword "type" shall be used to specify that each instance of the new notation contains at this point, using standard ASN.1 notation, some "Type".

A.3 *Macro definition notation*

A.3.1 A macro shall be defined using the notation "MacroDefinition":

```
MacroDefinition ::=
        macroreference
        MACRO
        "::="
        MacroSubstance

MacroSubstance ::=
        BEGIN MacroBody END |
        macroreference |
        Externalmacroreference

MacroBody ::=
        TypeProduction
        ValueProduction
        SupportingProductions

TypeProduction ::=
        TYPE NOTATION
        "::="
        MacroAlternativeList

ValueProduction ::=
        VALUE NOTATION
        "::="
        MacroAlternativeList
        .
SupportingProduction ::=
        ProductionList |
        empty

ProductionList ::=
        Production |
        ProductionList Production

Production ::=
        productionreference
        "::="
        MacroAlternativeList

Externalmacroreference ::=
        modulereference . macroreference
```

Note – It is intended that one macro definition be permitted to reference (i.e., use) other macros. Ensuring that the notation permits this is for further study.

A.3.2 If the "macroreference" alternative of "MacroSubstance" is chosen, then the module containing the macro definition shall either:

a) contain another macro definition defining that "macroreference"; or

b) contain the "macroreference" in its "SymbolsImported".

A.3.3 If the "Externalmacroreference" alternative of "MacroSubstance" is chosen, then the module denoted by "modulereference" shall contain a macro definition defining the "macroreference". The associated definition is then also associated with the "macroreference" being defined.

A.3.4 The chain of definitions which can arise from repeated applications of the rules of § A.3.2 to § A.3.3 shall terminate with a "MacroDefinition" which uses the "BEGIN MacroBody END" alternative and it is that "MacroBody" which defines the type and value notation for the macro being defined.

A.3.5 Each "productionreference" which occurs in a "SymbolDefn" (see § A.3.9) shall occur exactly once as the first item in a "Production".

A.3.6 Each instance of the new type notation shall commence with the sequence of characters in the "macroreference", followed by one of the sequences of characters referenced by "TYPE NOTATION" after applying the productions specified in the macro definition.

A.3.7 Each instance of the new value notation shall consist of one of the sequences of characters referenced by "VALUE NOTATION" after applying the productions specified in the macro definition.

A.3.8 The "MacroAlternativeList" in a production specifies the possible sets of character sequences referenced by the production. It is specified by:

```
MacroAlternative List ::=
        MacroAlternative |
        MacroAlternativeList " | " MacroAlternative
```

The set of character sequences referenced by the "MacroAlternativeList" consists of all the character sequences which are referenced by any of the "MacroAlternative" productions in the "MacroAlternativeList".

A.3.9 The notation for a "MacroAlternative" shall be:

```
MacroAlternative ::= SymbolList

SymbolList ::=
        SymbolElement                                   |
        SymbolList SymbolElement

SymbolElement ::=
        SymbolDefn                                      |
        EmbeddedDefinitions

SymbolDefn ::=
        astring                                         |
        productionreference                             |
        "string"                                        |
        "identifier"                                    |
        "number"                                        |
        "empty"                                         |
        type                                            |
        type(localtypereference)                        |
        value(MacroType)                                |
        value(localvaluereference MacroType)            |
        value(VALUE MacroType)

MacroType ::= localtypereference                        |
        Type
```

Note – When in a macro, any "MacroType" defined in that macro can appear at any point in which ASN.1 specifies a "Type".

A "MacroAlternative" references all characters strings which are formed by taking any of the character strings referenced by the first "SymbolDefn" in the "SymbolList", followed by any one of the character strings referenced by the second "SymbolDefn" in the "SymbolList", and so on, up to and including the last "SymbolDefn" in the "SymbolList".

Note – The "EmbeddedDefinitions" (if any) play no direct part in determining these strings.

A.3.10 An "astring" references the sequence of characters in the "astring" without the enclosing pair of ".

A.3.11 A "productionreference" references any sequence of characters specified by the "Production" it identifies.

A.3.12 The sequences of characters referenced by the next four alternatives for "SymbolDefn" are specified in Table 8/X.208.

Note – The sequences of characters referenced by the "string" should be terminated in an instance of the macro notation by the appearance of a sequence referenced by the next "SymbolDefn" in the "SymbolList".

A.3.13 A "type" references any sequence of symbols which forms a "Type" notation as specified in § 12.1.

Note – The "DefinedType" of § 12.1 may in this case contain a "localtypereference" referencing a type defined in the macro notation.

A.3.14 A "type(localtypereference)" references any sequence of symbols which forms a "Type" as specified in § 12.1, but in addition assigns that type to the "localtypereference". A later assignment may occur to the same "localtypereference".

A.3.15 A "value(MacroType)" references any sequence of symbols which forms a "Value" notation (as specified in § 12.7) for the type specified by the "Macro Type".

A.3.16 A "value(localvaluereference MacroType)" references any sequence of symbols which forms a "Value" notation (as specified in § 12.7) for the type specified by "MacroType", but in addition assigns the value specified by the value notation to the "localvaluereference". A later assignment may occur to the "localvaluereference".

A.3.17 A "value(VALUE MacroType)" references any sequence of symbols which forms a "Value" notation (as specified in § 12.7) for the type specified by "MacroType", but in addition returns the value as the value specified by the value notation. The type of the value returned is the type referenced by MacroType.

A.3.18 Precisely one assignment to VALUE (as specified in § A.3.17 or in § A.3.19) occurs in the analysis of any correct instance of the new value notation.

A.3.19 The notation for an "EmbeddedDefinitions" shall be:

```
EmbeddedDefinitions ::=
        <EmbeddedDefinitionList>

EmbeddedDefinitionList ::=
        EmbeddedDefinition |
        EmbeddedDefinitionList
        EmbeddedDefinition

EmbeddedDefinition ::=
        LocalTypeassignment |
        LocalValueassignment

LocalTypeassignment ::=
        localtypereference
        "::="
        MacroType

LocalValueassignment ::=
        localvaluereference
        MacroType
        "::="
        MacroValue

MacroValue ::=
        Value |
        localvaluereference
```

The assignment of a "MacroType" to a "localtypereference" (or of a "MacroValue" to a "localvaluereference") within an "EmbeddedDefinitions" takes effect during the syntax analysis of an instance of the new notation at the time when the "EmbeddedDefinitions" is encountered, and persists until redefinition of the "localtypereference" or "localvaluereference" occurs.

Note 1 – The use of the associated "localtypereference" or "localvaluereference" elsewhere in the "Alternative" implies assumptions on the nature of the parsing algorithm. Such assumptions should be indicated by comment. For example, use of the "localtypereference" textually following the "EmbeddedDefinitions" implies a left to right parse.

Note 2 – The "localvaluereference" "VALUE" may be assigned a value either by the "value (VALUE MacroType)" construct or by an "EmbeddedDefinition". In both cases, the value is returned, as specified in § A.3.17.

A.4 *Use of the new notation*

Whenever a "Type" (or "Value") notation is called for by this Recommendation, an instance of the type notation (or value notation) defined by a macro may be used, provided that the macro is either:

a) defined within the same module; or

b) imported into the module, by means of the appearance of the "macroreference" in the "SymbolsImported" of the module.

To allow the latter possibility, a "macroreference" can appear as a "Symbol" in § 9.1.

Note 1 – This extension to the standard ASN.1 notation is not shown in the body of this Recommendation.

Note 2 – It is possible to construct modules including sequences of type assignment and macro definitions which make parsing of the value syntax in DEFAULT values arbitrarily complex.

ANNEX B

(to Recommendation X.208)

ISO assignment of OBJECT IDENTIFIER component values

B.1 Three arcs are specified from the root node. The assignment of values and identifiers, and the authority for assignment of subsequent component values, are as follows:

Value	Identifier	Authority for subsequent assignments
0	ccitt	CCITT
1	iso	ISO
2	joint-iso-ccitt	See Annex D

Note – The remainder of this annex concerns itself only with ISO assignment of values.

B.2 The identifiers "ccitt", "iso" and "joint-iso-ccitt", assigned above, may each be used as a "NameForm".

B.3 Four arcs are specified from the node identified by "iso". The assignment of values and identifiers is

Value	Identifier	Authority for subsequent assignments
0	standard	See § B.4
1	registration-authority	See § B.5
2	member-body	See § B.6
3	identified-organization	See § B.7

These identifiers may be used as a "NameForm".

B.4 The arcs below "standard" shall each have the value of the number of an International Standard. Where the International Standard is multi-part, there shall be an additional arc for the part number, unless this is specifically excluded in the text of the International Standard. Further arcs shall have values as defined in that International Standard.

Note – If a non-multipart International Standard allocates object identifiers, and subsequently becomes a multipart International Standard, it shall continue to allocate object identifiers as if it were a single part International Standard.

B.5 The arcs below "registration authority" are reserved for an addendum to this Recommendation which will be progressed alongide the establishment of procedures for the identification of specific OSI Registration Authorities.

B.6 The arcs immediately below "member-body" shall have values of three digit numeric country code, as specified in ISO 3166, that identifies the ISO Member Body in that country (see Note). The "NameForm" of object identifier component is not permitted with these identifiers. Arcs below the "country code" are not defined in this Recommendation.

Note – The existence of a country code in ISO 3166 does not necessarily imply that there is an ISO Member Body representing that country or that the ISO Member Body for that country administers a scheme for the allocation of object identifier components.

B.7 The arcs immediately below "identified-organization" shall have values of an International Code Designator (ICD) allocated by the Registration Authority for ISO 6523 that identify an issuing organization specifically registered by the authority as allocating object identifier components (see Notes 1 and 2). The arcs immediately below the ICD shall have values of an "organization code" allocated by the issuing organization in accordance with ISO 6523. Arcs below "organization code" are not defined by this Recommendation (see Note 3).

Note 1 – The requirement that issuing organizations are recorded by the Registration Authority for ISO 6523 as allocating organization codes for the purpose of object identifier components ensures that only numerical values in accordance with this Recommendation are allocated.

Note 2 – The declaration that an issuing organization allocates organization codes for the purpose of object identifier components does not preclude the use of these codes for other purposes.

Note 3 – It is assumed that the organizations identified by the "organization code" will define further arcs in such a way as to ensure allocation of unique values.

Note 4 – The effect of B.7 is that any organization can obtain an organization code from an appropriate issuing organization, and can then assign OBJECT IDENTIFIER values for its own purposes, with the assurance that those values will not conflict with values assigned by other organizations. By this means, a manufacturer could, for example, assign an OBJECT IDENTIFIER to its own proprietary information formats.

ANNEX C

(to Recommendation X.208)

CCITT assignment of OBJECT IDENTIFIER component values

C.1 Three arcs are specified from the root node. The assignment of values and identifiers, and the authority for assignment of subsequent component values, are as follows:

Value	Identifier	Authority for subsequent assignments
0	ccitt	CCITT
1	iso	ISO
2	joint-iso-ccitt	See Annex D

Note – The remainder of this annex concerns itself only with CCITT assignment of values.

C.2 The identifiers "ccitt", "iso" and "joint-iso-ccitt", assigned above, may each be used as a "NameForm".

C.3 Four arcs are specified from the node identified by "ccitt". The assignment of values and identifiers is

Value	Identifier	Authority for subsequent assignments
0	recommendation	See § C.4
1	question	See § C.5
2	administration	See § C.6
3	network-operator	See § C.7

These identifiers may be used as a "NameForm".

C.4 The arcs below "recommendation" have the value 1 to 26 with assigned identifiers of a to z. Arcs below these have the numbers of CCITT Recommendations in the series identified by the letter. Arcs below this are determined as necessary by the CCITT Recommendation. The identifiers a to z may be used as a "NameForm".

C.5 The arcs below "question" have values corresponding to CCITT Study Groups, qualified by the Study Period. The value is computed by the formula:

study group number + (Period * 32)

where "Period" has the value 0 for 1984-1988, 1 for 1988-1992, etc., and the multiplier is 32 decimal.

The arcs below each study group have the values corresponding to the questions assigned to that study group. Arcs below this are determined as necessary by the group (e.g. working party or special rapporteur group) assigned to study the question.

C.6 The arcs below administration have the values of X.121 DCCs. Arcs below this are determined as necessary by the Administration of the country identified by the X.121 DCC.

C.7 The arcs below "network-operator" have the value of X.121 DNICs. Arcs below this are determined as necessary by the Administration or RPOA identified by the DNIC.

ANNEX D

(to Recommendation X.208)

Joint assignment of OBJECT IDENTIFIER component values

D.1 Three arcs are specified from the root node. The assignment of values and identifiers, and the authority for assignment of subsequent component values, are as follows:

Value	Identifier	Authority for subsequent assignments
0	ccitt	CCITT
1	iso	ISO
2	joint-iso-ccitt	See below

Note – The remainder of this annex concerns itself only with ISO-CCITT assignment of values.

D.2 The identifiers "ccitt", "iso" and "joint-iso-ccitt", assigned above, may each be used as a "NameForm".

D.3 The arcs below "joint-iso-ccitt" have values which are assigned and agreed from time to time by ISO and CCITT to identify areas of joint ISO-CCITT standardisation activity, in accordance with the "Procedures for assignment of object identifier component values for joint ISO-CCITT use" [1].

D.4 The arcs beneath each arc identified by the mechanisms of § D.3 shall be allocated in accordance with mechanisms established when the arc is allocated.

Note – It is expected that this will involve delegation of authority to the joint agreement of CCITT and ISO Rapporteurs for the joint area of work.

D.5 Initial ISO Standards and CCITT Recommendations in areas of joint ISO-CCITT activity require to allocate OBJECT IDENTIFIERS in advance of the establishment of the procedures of D.3, and hence allocate in accordance with annexes B or C. Information objects identified in this way by ISO Standards or CCITT Recommendations shall not have their OBJECT IDENTIFIERS changed when the procedures of D.3 are established.

---

[1] The Registration Authority for assignment of object identifier component values for joint ISO-CCITT use is the American National Standards Institute (ANSI), 1430 Broadway, New York, NY 10018, USA.

APPENDIX I

(to Recommendation X.208)

Exemples and hints

This appendix contains examples of the use of ASN.1 in the description of (hypothetical) data structures. It also contains hints, or guidelines, for the use of the various features of ASN.1.

I.1 *Example of a personnel record*

The use of ASN.1 is illustrated by means of a simple, hypothetical personnel record.

I.1.1 *Informal description of personnel record*

The structure of the personnel record and its value for a particular individual are shown below.

Name:	John P Smith
Title:	Director
Employee Number:	51
Date of Hire:	17 September 1971
Name of Spouse:	Mary T Smith
Number of Children:	2
Child Information	
Name:	Ralph T Smith
Date of Birth	11 November 1957
Child Information	
Name:	Susan B Jones
Date of Birth	17 July 1959

I.1.2 *ASN.1 description of the record structure*

The structure of every personnel record is formally described below using the standard notation for data types.

```
PersonnelRecord ::= [APPLICATION 0] IMPLICIT SET
{                                   Name,
 title                              [0] VisibleString,
 number                             EmployeeNumber,
 dateOfHire                         [1] Date,
 nameOfSpouse                       [2] Name,
 children                           [3] IMPLICIT
                                    SEQUENCE OF
                                    ChildInformation
                                    DEFAULT {}  }

ChildInformation ::= SET
    {                               Name,
     dateOfBirth                    [0] Date}

Name ::= [APPLICATION 1] IMPLICIT SEQUENCE
{givenName                          VisibleString,
 initial                            VisibleString,
 familyName                         VisibleString}

EmployeeNumber ::= [APPLICATION 2] IMPLICIT INTEGER

Date ::= [APPLICATION 3] IMPLICIT VisibleString-- YYYYMMDD
```

This example illustrates an aspect of the parsing of the ASN.1 syntax. The sintactic construct "DEFAULT" can only be applied to an element of a "SEQUENCE" or a "SET", it cannot be applied to an element of a "SEQUENCE OF". Thus the "DEFAULT { }" in "PersonnelRecord" applies to "children", not to "ChildInformation".

I.1.3 *ASN.1 description of a record value*

The value of John Smith's personnel record is formally described below using the standard notation for data values.

```
{                                {givenName "John",initial "P",familyName "Smith"}
 title                           "Director"
 number                          51
 dataOfHire                      "19710917"
 nameOfSpouse                    {givenName "Mary",initial "T",familyName "Smith"}
 children
   {{{ givenName "Ralph",initial "T",familyName "Smith"}
         dateOfBirth "19571111"}
         {{givenName "Susan",initial "B",familyName "Jones"}
         dateOfBirth "19590717"}}}
```

I.2 *Guidelines for use of the notation*

The data types and formal notation defined by this Recommendation are flexible, allowing a wide range of protocols to be designed using them. This flexibility, however, can sometimes lead to confusion, especially when the notation is approached for the first time. This Annex attempts to minimise confusion by giving guidelines for, and examples of, the use of the notation. For each of the built-in data types, one or more usage guidelines are offered. The character string types (for example, VisibleString) and the types defines in section three are not dealt with here.

I.2.1 *Boolean*

I.2.1.1 Use a Boolean type to model the values of a logical (that is, two-state) variable, for example, the answer to a yes-or-no question.

Example:

```
Employed ::= BOOLEAN
```

I.2.1.2 When assigning a reference name to a Boolean type, choose one that describes the **true** state.

Example:

```
Married ::= BOOLEAN

not

MaritalStatus ::= BOOLEAN
```

See also § I.2.3.2

I.2.2 *Integer*

I.2.2.1 Use an integer type to model the values (for all practical purposes, unlimited in magnitude) of a cardinal or integer variable.

Example:

```
CheckingAccountBalance ::= INTEGER
  -- in cents; negative means overdrawn
```

I.2.2.2 Define the minimum and maximum allowed values of an integer type as distinguished values.

Example:

```
DayOfTheMonth ::= INTEGER {first(1),last(31)}
```

I.2.3 *Enumerated*

I.2.3.1 Use an enumerated type with distinguished values to model the values of a variable with three or more states. Assign values starting with zero if their only constraint is distinctness.

Example:

```
DayOfTheWeek ::= ENUMERATED
  {sunday(0),monday(1),tuesday(2),wednesday(3),thursday(4),friday(5),saturday(6)}
```

I.2.3.2 Use an enumerated type to model the values of a variable that has just two states now but that may have additional states in a future version of the protocol.

Example:

```
MaritalStatus ::= ENUMERATED {single(0),married(1)}
```

in anticipation of

```
MaritalStatus ::= ENUMERATED {single(0),married(1),widowed(2)}
```

I.2.4 *Real*

I.2.4.1 Use a real type to model an approximate number.

Example:

```
AngleInRadians ::= REAL
pi REAL ::= {3141592653589793238462643383279, 10, -30}
```

I.2.5 *Bit string*

I.2.5.1 Use a bit string type to model binary data whose format and length are unspecified, or specified elsewhere, and whose length in bits is not necessarily a multiple of eight.

Example:

```
G3FacsimilePage ::= BIT STRING
  -- a sequence of bits conforming to CCITT
  -- Recommendation T.4.
```

I.2.5.2 Define the first and last meaningful bits of a fixed-length bit string as distinguished bits.

Example:

```
Nibble ::= BIT STRING {first(0),last(3)}
```

I.2.5.3 Use a bit string type to model the values of a **bit map**, an ordered collection of logical variables indicating whether a particular condition holds for each of a correspondingly ordered collection of objects.

Example:

```
SunnyDaysOfTheMonth ::= BIT STRING {first(1),last(31)}
  -- Day i was sunny if and only if bit i is one
```

I.2.5.4 Use a bit string type with distinguished values to model the values of a collection of related logical variables.

Example:

```
PersonalStatus ::= BIT STRING
  {married(0),employed(1),veteran(2),collegeGraduate(3)}
```

I.2.6 *Octet string*

I.2.6.1 Use an octet string type to model binary data whose format and length are unspecified, or specified elsewhere, and whose length in bits is a multiple of eight.

Example:

```
G4FacsimileImage ::= OCTET STRING
  -- a sequence of octets conforming to
  -- CCITT Recommendations T.5 and T.6
```

I.2.6.2 Use a character string type in preference to an octet string type, where an appropriate one is available.

Example:

```
Surname ::= PrintableString
```

I.2.6.3 Use an octet string type to model any string of information which cannot be modelled using one of the character string types. Be sure to specify the repertoire of characters and their coding into octets.

Example:

```
PackedBCDString ::= OCTET STRING
  -- the digits 0 through 9, two digits per octet,
  -- each digit encoded as 0000 to 1001,
  -- 1111₂ used for padding.
```

I.2.7 *Null*

Use a null type to indicate the effective absence of an element of a sequence.

Example:

```
PatientIdentifier ::= SEQUENCE
 {name                    VisibleString,
  roomNumber              CHOICE
          {INTEGER,
          NULL -- if an out-patient --}}
```

Note – The use of "OPTIONAL" provides an equivalent facility.

I.2.8 *Sequence and sequence-of*

I.2.8.1 Use a sequence-of type to model a collection of variables whose types are the same, whose number is large or unpredictable, and whose order is significant.

Example:

```
NamesOfMemberNations ::= SEQUENCE OF VisibleString
  -- in the order in which they joined
```

I.2.8.2 Use a sequence type to model a collection of variables whose types are the same, whose number is known and modest, and whose order is significant, provided that the makeup of the collection is unlikely to change from one version of the protocol to the next.

Example:

```
NamesOfOfficers ::= SEQUENCE
{president                VisibleString,

 vicePresident            VisibleString,

 secretary                VisibleString}
```

I.2.8.3 Use a sequence type to model a collection of variables whose types differ, whose number is known and modest, and whose order is significant, provided that the makeup of the collection is unlikely to change from one version of the protocol to the next.

Example:

```
Credentials ::= SEQUENCE
 {userName                VisibleString,
  password                VisibleString,
  accountNumber           INTEGER}
```

I.2.8.4 If the elements of a sequence type are fixed in number but of several types, a reference name should be assigned to every element whose purpose is not fully evident from its type.

Example:

```
File ::= SEQUENCE
 {                          ContentType,
  other                     FileAttributes,
  content                   ANY}
```

See also § I.2.5.3, § I.2.5.4, and § I.2.7.

I.2.9 *Set*

I.2.9.1 Use a set type to model a collection of variables whose number is known and modest and whose order is insignificant. Identify each variable by context-specifically tagging it.

Example:

```
UserName ::= SET
 {personalName              [0] IMPLICIT VisibleString,
  organisationName          [1] IMPLICIT VisibleString,
  countryName               [2] IMPLICIT VisibleString}
```

I.2.9.2 Use a set type with "OPTIONAL" to model a collection of variables that is a (proper or improper) subset of another collection of variables whose number is known and reasonably small and whose order is insignificant. Identify each variable by context-specifically tagging it.

Example:

```
UserName ::= SET
 {personalName              [0] IMPLICIT VisibleString,
  organisationName          [1] IMPLICIT VisibleString OPTIONAL
    -- defaults to that of the local organisation --,

  countryName               [2] IMPLICIT VisibleString OPTIONAL
    -- defaults to that of the local country --}
```

I.2.9.3 Use a set type to model a collection of variables whose makeup is likely to change from one version of the protocol to the next. Identify each variable by context-specifically tagging it.

Example:

```
UserName ::= SET
 {personalName              [0] IMPLICIT VisibleString,
  organisationName          [1] IMPLICIT VisibleString OPTIONAL
    -- defaults to that of the local organisation --,

  countryName               [2] IMPLICIT VisibleString OPTIONAL
    -- defaults to that of the local country
    -- other optional attributes are for further study --}
```

I.2.9.4 If the members of a set type are fixed in number, a reference name should be assigned to every member whose purpose is not fully evident from its type.

Example:

```
FileAttributes ::= SET
 {owner                     [0] IMPLICIT UserName,
  sizeOfContentInOctets     [1] IMPLICIT INTEGER,
                            [2] IMPLICIT AccessControls,
  ...}
```

I.2.9.5 Use a set type to model a collection of variables whose types are the same and whose order is insignificant.

Example:

```
Keywords ::= SET OF VisibleString -- in arbitrary order
```

See also § I.2.5.4 and § I.2.10.3.

I.2.10 *Tagged*

I.2.10.1 Use a universal tagged type to define – in this Recommendation only – a generally useful, application independent data type that must be distinguishable (by means of its representation) from all other data types.

Example:

```
EncryptionKey ::= [UNIVERSAL 30] IMPLICIT OCTET STRING
-- seven octets
```

I.2.10.2 Use an application-wide tagged type to define a data type that finds wide, scattered use within a particular presentation context and that must be dintinguisable (by means of its representation) from all other data types used in the presentation context.

Example:

```
FileName ::= [APPLICATION 8] IMPLICIT SEQUENCE
 {directoryName              VisibleString,
  directoryRelativeFileName VisibleString}
```

I.2.10.3 Use context-specific tagged types to distinguish the members of a set. Assign numeric tags starting with zero if their only constraint is distinctness.

Example:

```
CustomerRecord ::= SET
 {name                       [0] IMPLICIT VisibleString,
  mailingAddress             [1] IMPLICIT VisibleString,
  accountNumber              [2] IMPLICIT INTEGER,
  balanceDue                 [3] IMPLICIT INTEGER -- in cents --}
```

I.2.10.4 Where a particular set member has been application-wide tagged, a further context-specific tag need not be used, unless it is (or may be in the future) needed for distinctness. Where the set member has been universally tagged, a further context-specific tag should be used.

Example:

```
ProductRecord ::= SET
 {                           UniformCode,
  description                [0] IMPLICIT VisibleString,
  inventoryNo                [1] IMPLICIT INTEGER,
  inventoryLevel             [2] IMPLICIT INTEGER}

UniformCode ::= [APPLICATION 13] IMPLICIT INTEGER
```

I.2.10.5 Use context-specific tagged types to distinguish the alternatives of a CHOICE. Assign numeric tags starting with zero if their only constraint is distinctness.

Example:

```
CustomerAttribute ::= CHOICE
 {name                [0] IMPLICIT VisibleString,
  mailingAddress      [1] IMPLICIT VisibleString,
  accountNumber       [2] IMPLICIT INTEGER,
  balanceDue          [3] IMPLICIT INTEGER -- in cents --}
```

I.2.10.6 Where a particular CHOICE alternative has been defined using an application-wide tagged type, a further context-specific tag need not be used, unless it is (or maybe in the future) needed for distinctness.

Example:

```
ProductDesignator ::= CHOICE
 {                    UniformCode,
  description         [0] IMPLICIT VisibleString,
  inventoryNo         [1] IMPLICIT INTEGER}

UniformCode ::= [APPLICATION 13] IMPLICIT INTEGER
```

I.2.10.7 Where a particular CHOICE alternative has been universally tagged, a further context-specific tag should be used, unless the provision of more than one universal type is the purpose of the choice.

Example:

```
CustomerIdentifier ::= CHOICE
 {name                VisibleString,
  number              INTEGER}
```

I.2.10.8 Use a private-use tagged type to define a data type that finds use within a particular organisation or country and that must be distinguishable (by means of its representation) from all other data types used by that organisation or country.

Example:

```
AcmeBadgeNumber ::= [PRIVATE 2] IMPLICIT INTEGER
```

I.2.10.9 These guidelines use implicit tagging in the examples whenever it is legal to do so. This may, depending on the encoding rules, result in a compact representation, which is highly desirable in some applications. In other applications, compactness may be less important than, for example, the ability to carry out strong type-checking. In the latter case, explicit tagging can be used.

See also § I.2.9.1, § I.2.9.2, § I.2.11.1 and § I.2.11.2.

I.2.11 *Choice*

I.2.11.1 Use a CHOICE to model a variable that is selected from a collection of variables whose number is known and modest. Identify each candidate variable by context-specifically tagging it.

Example:

```
FileIdentifier ::= CHOICE
 {relativeName        [0] IMPLICIT VisibleString,
     -- name of file (for example, "MarchProgressReport")

  absoluteName        [1] IMPLICIT VisibleString,
     -- name of file and containing directory
     -- (for example, "<Williams>MarchProgressReport")

  serialNumber        [2] IMPLICIT INTEGER
     -- system-assigned identifier for file --}
```

I.2.11.2 Use a CHOICE to model a variable that is selected from a collection of variables whose makeup is likely to change from one version of the protocol to the next. Identify each candidate variable by context-specifically tagging it.

Example:

```
FileIdentifier ::= CHOICE
 {relativeName              [0] IMPLICIT VisibleString,
     -- name of file (for example, "MarchProgressReport")

  absoluteName              [1] IMPLICIT VisibleString,
     -- name of file and containing directory
     -- (for example, "<Williams>MarchProgressReport")
  -- other forms of file identifiers are for further study --}
```

I.2.11.3 A reference name should be assigned to each alternative whose purpose is not fully evident from its type.

Example:

```
FileIdentifier ::= CHOICE
 {relativeName              [0] IMPLICIT VisibleString,
     -- name of file (for example, "MarchProgressReport")

  absoluteName              [1] IMPLICIT VisibleString,
     -- names of file and containing directory
     -- (for example, "<Williams>MarchProgressReport")

                            [2] IMPLICIT SerialNumber
     -- system-assigned identifier for file --}
```

I.2.11.4 Where implicit tagging is the norm in a particular application of this Recommendation, use a CHOICE of only one type where the possibility is envisaged or more than one type being permitted in the future. This precludes the possibility of implicit tagging taking place, and thus aids transition.

Example:

```
Greeting ::= [APPLICATION 12] CHOICE
                            {VisibleString}
```

in anticipation of

```
Greeting ::= [APPLICATION 12] CHOICE
                            {VisibleString,
                            Voice}
```

I.2.12 *Selection type*

I.2.12.1 Use a selection type to model a variable whose type is that of some particular alternatives of a previously defined CHOICE.

I.2.12.2 Consider the definition:

```
FileAttribute ::= CHOICE
{date-last-used             INTEGER,
 file-name                  VisibleString}
```

then the following definition is possible:

```
CurrentAttributes ::= SEQUENCE
{date-last-used             <FileAttribute,
 file-name                  <FileAttribute}
```

with a possible value notation of

```
{date-last-used 27
 file-name "PROGRAM"}
```

The following definition is also possible:

```
AttributeList ::= SEQUENCE
{first-attribute date-last-used   <FileAttribute,
 second-attribute file-name       <FileAttribute}
```

with a possible value notation of

```
{first-attribute 27,
 second-attribute "PROGRAM"}
```

I.2.13 *Any*

I.2.13.1 Use an any type to model a variable whose type is unspecified, or specified elsewhere using ASN.1.

Example:

```
MessageContents ::= ANY
  -- a data element whose type is specified
  -- outside this Recommendation using the ASN.1 notation.
```

I.2.14 *External*

I.2.14.1 Use an external type to model a variable whose type is unspecified, or specified elsewhere with no restriction on the notation used to specify the type.

Example:

```
FileContents ::= EXTERNAL

DocumentList ::= SEQUENCE OF EXTERNAL
```

I.3 *An example of the use of the macro notation*

Suppose it is desired to have a notation for type definition of the form

PAIR TYPEX = ... TYPEY = ...

with a corresponding value notation allowing

(X = ----, Y = ----)

The .... and the ---- refer to any ASN.1 type and corresponding value respectively.

This macro type notation could be used to define types and values as follows:

```
T1 ::= PAIR
        TYPEX = INTEGER
        TYPEY = BOOLEAN

T2 ::= PAIR
        TYPEX = VisibleString
        TYPEY = T1
```

Then a value of type T1 might be:

(X = 3, Y = TRUE)

and a value of type T2 might be:

(X = "Name", Y = (X = 4, Y = FALSE))

The following macro definition could be used to establish this new notation, as an extension of the basic ASN.1:

```
PAIR
MACRO ::= BEGIN
TYPE NOTATION ::=
        "TYPEX"
        "="
        type (Local-type-1)
          -- Expects any ASN.1 type and assigns it
          -- to the variable Local-type-1;
        "TYPEY"
        "="
        type (Local-type-2)
          -- Expects a second ASN.1 type and assigns
          -- it to the variable Local-type-2;

VALUE NOTATION ::=
        "("
        "X"
        "="
        value (Local-value-1 Local-type-1)
          -- Expects a value for the type in
          -- Local-type-1, and assigns it
          -- to the variable Local-value-1;
        ","
        "Y"
        "="
        value (Local-value-2 Local-type-2)
          -- Expects a value for the type in
          -- Local-type-2 and assigns it
          -- to the variable Local-value-2;
        <VALUE SEQUENCE {Local-type-1, Local-type-2}
          ::= {Local-value-1, Local-value-2}>
        -- This "embedded definition" returns
        -- the final value as the value
        -- of a sequence of the two types.
        ")"
END
```

In this example, the basic ASN.1 type of the returned value is independent of the actual instance of the value notation, but does depend on the instance of the type notation that is used. In other cases, it may be fully determined by the macro definition, or it may depend on the instance of the value notation that is used. Note, however, that in all cases it is the "VALUE NOTATION" production that needs to be examined in order to determine the type of the returned value. The "TYPE NOTATION" production simply defines the syntax for type definition, and establishes the initial value of local variables for use when analysing an instance of the value notation.

I.4 *Use in identifying abstract syntaxes*

I.4.1 Use of the presentation service (Recommendation X.216) requires the specification of values called **presentation data values** and the grouping of those presentation data values into sets which are called **abstract syntaxes**. Each of these sets is given an **abstract syntax name** of ASN.1 type object identifier.

I.4.2 ASN.1 can be used as a general tool in the specification of presentation data values and their grouping into named abstract syntaxes.

I.4.3 In the simplest such use, there is a single ASN.1 type such that every presentation data value in the named abstract syntax is a value of that ASN.1 type. This type will normally be a choice type, and every presentation data value will be an alternative type from this choice type. In this case it is recommended that the ASN.1 module notation be used to contain this choice type as the first defined type, followed by the definition of those (non-universal) types referenced directly or indirectly by this choice type.

Note – This is not intended to exclude references to types defined in other modules.

I.4.4 The following is an example of text which might appear in an application standard. The end of the example is identified by the phrase "END OF EXAMPLE" in order to avoid confusion.

Example:

```
ISOxxxx-yyyy DEFINITIONS ::=
BEGIN
PDU ::= CHOICE
    {connect-pdu      ...,
     data-pdu      CHOICE
                 {  ...,
                ...}     ,
    ...              }
...
END
```

This International Standard assigns the ASN.1 object identifier value

{iso standard xxxx abstract-syntax (1)}

as an abstract syntax name for the set of presentation data values, each of which is a value of the ASN.1 type "ISOxxxx-yyyy.PDU".

The corresponding ASN.1 object descriptor value shall be

"....."

The ASN.1 object identifier and object descriptor values

{joint-iso-ccitt asn1 (1) basic-encoding (1)}

and

"Basic Encoding of a single ASN.1 type"

(assigned to an information object in Recommendation X.209) can be used as a transfer syntax name with this abstract syntax name.

END OF EXAMPLE

I.4.5 The standard may additionally make support of the transfer syntax obtained by applying

{joint-iso-ccitt asn1 (1) basic-encoding (1)}

mandatory for this abstract syntax

I.5 *Subtypes*

I.5.1 Use subtypes to limit the values of an existing type which are to be permitted in a particular situation.

Examples:

```
AtomicNumber ::= INTEGER (1..104)

TouchToneString ::= IA5String (FROM
                              ("0"|"1"|"2"|"3"|"4"|"5"|"6"|
                              "7"|"8"|"9"|"*"|"#")|
                              SIZE (1..63))

ParameterList ::= SET SIZE (0..63) OF Parameter

SmallPrime ::= INTEGER (2|3|5|7|11|13|17|19|23|29)
```

I.5.2 Where two or more related types have significant commonality, consider explicitly defining their common parent as a type and use subtyping for the individual types. This approach makes clear the relationship and the commonality, and encourages (though does not force) this to continue as the types evolve. It thus facilitates the use of common implementation approaches to the handling of values of these types.

Example:

```
Envelope ::= SET {typeA TypeA,
                  typeB TypeB OPTIONAL,
                  typeC TypeC OPTIONAL}
                  -- the common parent

ABEnvelope ::= Envelope (WITH COMPONENTS
                  {...,
                   typeB PRESENT, typeC ABSENT})
                  -- where typeB must always
                  -- appear and typeC must not

ACEnvelope ::= Envelope (WITH COMPONENTS
                  {...,
                   typeB ABSENT, typeC PRESENT})
                  -- where typeC must always
                  -- appear and typeB must not
```

The latter definitions could alternatively be expressed as

```
ABEnvelope ::= Envelope (WITH COMPONENTS {typeA,typeB})

ACEnvelope ::= Envelope (WITH COMPONENTS {typeA,typeC})
```

The choice between the alternatives would be made upon such factors as the number of components in the parent type, and the number of those which are optional, the extent of the difference between the individual types, and the likely evolution strategy.

I.5.3 Use subtyping to partially define a value, for example, a protocol data unit to be tested for in a conformance test, where the test is concerned only with some components of the PDU.

Example:

Given:

```
PDU ::= SET
          {alpha    [0]  INTEGER,
           beta     [1]  IA5String OPTIONAL,
           gamma    [2]  SEQUENCE OF Parameter,
           delta    [3]  BOOLEAN}
```

then in composing a test which requires the Boolean to be false and the integer to be negative, write:

```
TestPDU ::= PDU (WITH COMPONENTS
                  {...,
                   delta (FALSE),
                   alpha (MIN...<0)})
```

and if, further, the IA5String, beta, is to be present and either 5 or 12 characters in length, write:

```
FurtherTestPDU ::= TestPDU (WITH COMPONENTS
                  {...,
                   beta (SIZE (5|12)) PRESENT})
```

I.5.4 If a general-purpose datatype has been defined as a SEQUENCE OF, use subtyping to define a restricted subtype of the general type:

Example:

```
Text-block ::= SEQUENCE OF VisibleString
Address ::= Text-block
                    (SIZE (1..6) |
                    WITH COMPONENT (SIZE(1..32)))
```

I.5.5 Use contained subtypes to form new subtypes from existing subtypes:

Example:

```
Months ::= ENUMERATED {
                    january (1),
                    february (2),
                    march (3),
                    april (4),
                    may (5),
                    june (6),
                    july (7),
                    august (8),
                    september (9),
                    october (10),
                    november (11),
                    december (12)}

First-quarter ::= Months (
                    january |
                    february |
                    march)

Second-quarter ::= Months (
                    april |
                    may |
                    june)

Third-quarter ::= Months (
                    july |
                    august |
                    september)

Fourth-quarter ::= Months (
                    october |
                    november |
                    december)

First-half ::= Months (
                    INCLUDES First-quarter |
                    INCLUDES Second-quarter)

Second-half ::= Months (
                    INCLUDES Third-quarter |
                    INCLUDES Fourth-quarter )
```

APPENDIX II

(to Recommendation X.208)

Summary of the ASN.1 notation

The following items are defined in clause 8:

typereference	INTEGER	BEGIN
identifier	BIT	END
valuereference	STRING	DEFINITIONS
modulereference	OCTET	EXPLICIT
comment	NULL	ENUMERATED
empty	SEQUENCE	EXPORTS
number	OF	IMPORTS
bstring	SET	
hstring	IMPLICIT	REAL
cstring	CHOICE	INCLUDES
"::="	ANY	MIN
{	EXTERNAL	MAX
}	OBJECT	SIZE
<	IDENTIFIER	FROM
,	OPTIONAL	WITH
.	DEFAULT	COMPONENT
(	COMPONENTS	PRESENT
)	UNIVERSAL	ABSENT
[	APPLICATION	DEFINED
]	PRIVATE	BY
-	TRUE	PLUS-INFINITY
BOOLEAN	FALSE	MINUS-INFINITY

The following productions are used in this Recommendation, with the above items as terminal symbols:

```
ModuleDefinition ::=
        ModuleIdentifier
        DEFINITIONS
        TagDefault
        "::="
        BEGIN
        ModuleBody
        END

TagDefault ::=
        EXPLICIT TAGS |
        IMPLICIT TAGS |
        empty

ModuleIdentifier ::=
        modulereference
        AssignedIdentifier

AssignedIdentifier ::=
        ObjectIdentifier Value |
        empty

ModuleBody ::=
        Exports Imports AssignmentList |
        empty

Exports ::=
        EXPORTS SymbolsExported; |
        empty

SymbolsExported ::=
        SymbolList |
        empty
```

```
Imports ::=
        IMPORTS SymbolsImported; |
        empty

SymbolsImported ::=
        SymbolsFromModuleList |
        empty

SymbolsFromModuleList ::=
        SymbolsFromModule SymbolsFromModuleList |
        SymbolsFromModule

SymbolsFromModule ::=
        SymbolList FROM ModuleIdentifier

SymbolList ::= Symbol, SymbolList | Symbol

Symbol ::= typereference | valuereference

AssignmentList ::=
        Assignment AssignmentList |
        Assignment

Assignment ::= Typeassignment | Valueassignment

Externaltypereference ::=
        modulereference
        .
        typereference

Externalvaluereference ::=
        modulereference
        .
        valuereference

DefinedType ::= Externaltypereference | typereference

DefinedValue ::= Externalvaluereference | valuereference

Typeassignment ::=
        typereference
        "::="
        Type

Valueassignment ::=
        valuereference
        Type
        "::="
        Value

Type ::= BuiltinType | DefinedType | Subtype

BuiltinType ::=
        BooleanType             |
        IntegerType             |
        BitStringType           |
        OctetStringType         |
        NullType                |
        SequenceType            |
        SequenceOfType          |
        SetType                 |
        SetOfType               |
        ChoiceType              |
        SelectionType           |
        TaggedType              |
        AnyType                 |
        ObjectIdentifierType    |
        CharacterStringType     |
        UsefulType              |
        EnumeratedType          |
        RealType                |
```

```
NamedType ::= identifier Type | Type | SelectionType

Value ::= BuiltinValue | DefinedValue
BuiltinValue ::=
        BooleanValue          |
        IntegerValue          |
        BitStringValue        |
        OctetStringValue      |
        NullValue             |
        SequenceValue         |
        SequenceOfValue       |
        SetValue              |
        SetOfValue            |
        ChoiceValue           |
        SelectionValue        |
        TaggedValue           |
        AnyValue              |
        ObjectIdentifierValue |
        CharacterStringValue  |
        EnumeratedValue       |
        RealValue             |

NamedValue ::= identifier Value | Value

BooleanType ::= BOOLEAN

BooleanValue ::= TRUE | FALSE

IntegerType ::= INTEGER | INTEGER {NamedNumberList}

NamedNumberList ::=
        NamedNumber |
        NamedNumberList,NamedNumber

NamedNumber ::=
        identifier(SignedNumber) |
        identifier(DefinedValue)

SignedNumber ::= number | -number

IntegerValue ::= SignedNumber | identifier

EnumeratedType ::= ENUMERATED {Enumeration}

Enumeration ::=
        NamedNumber |
        NamedNumber, Enumeration

EnumeratedValue ::= identifier

RealType ::= REAL

RealValue ::= NumericRealValue | SpecialRealValue

NumericRealValue ::= {Mantissa, Base, Exponent} | 0

Mantissa ::= SignedNumber

Base ::= 2 | 10

Exponent ::= SignedNumber

SpecialRealValue ::= PLUS-INFINITY | MINUS-INFINITY

BitStringType ::= BIT STRING | BIT STRING {NamedBitList}

NamedBitList ::= NamedBit | NamedBitList,NamedBit

NamedBit ::=
        identifier(number) |
        identifier(DefinedValue)
```

```
BitStringValue ::= bstring | hstring |{IdentifierList} | {}

IdentifierList ::= identifier | IdentifierList,identifier

OctetStringType ::= OCTET STRING

OctetStringValue ::= bstring | hstring

NullType ::= NULL

NullValue ::= NULL

SequenceType ::=
        SEQUENCE {ElementTypeList} |
        SEQUENCE {}

ElementTypeList ::=
        ElementType |
        ElementTypeList,ElementType

ElementType ::=
        NamedType |
        NamedType OPTIONAL |
        NamedType DEFAULT Value |
        COMPONENTS OF Type

SequenceValue ::= {ElementValueList} | {}

ElementValueList ::=
        NamedValue |
        ElementValueList,NamedValue

SequenceOfType ::= SEQUENCE OF Type | SEQUENCE

SequenceOfValue ::= {ValueList} | {}

ValueList ::= Value | ValueList,Value

SetType ::= SET{ElementTypeList} | SET {}

SetValue ::= {ElementValueList} | {}

SetOfType ::= SET OF Type | SET

SetOfValue ::= {ValueList} | {}

ChoiceType ::= CHOICE{AlternativeTypeList}

AlternativeTypeList ::=
        NamedType |
        AlternativeTypeList,NamedType

ChoiceValue ::= NamedValue

SelectionType ::= identifier < Type

SelectionValue ::= NamedValue

TaggedType ::=
        Tag Type |
        Tag IMPLICIT Type |
        Tag EXPLICIT Type

Tag ::= [Class ClassNumber]

ClassNumber ::= number | DefinedValue

Class ::=
        UNIVERSAL |
        APPLICATION |
        PRIVATE |
        empty
```

```
TaggedValue ::= Value

AnyType ::=
        ANY |
        ANY DEFINED BY identifier

AnyValue ::= Type Value

ObjectIdentifierType ::= OBJECT IDENTIFIER

ObjectIdentifierValue ::=
        {ObjIdComponentList} |
        {DefinedValue ObjIdComponentList}

ObjIdComponentList ::=
        ObjIdComponent |
        ObjIdComponent ObjIdComponentList

ObjIdComponent ::=
        NameForm |
        NumberForm |
        NameAndNumberForm

NameForm ::= identifier

NumberForm ::= number | DefinedValue

NameAndNumberForm ::= identifier(NumberForm)

CharacterStringType ::= typereference

CharacterStringValue ::= cstring

UsefulType ::= typereference
```

The following characterstring types are defined in section two:

NumericString	VisibleString
PrintableString	ISO646String
TeletexString	IA5String
T61String	GraphicString
VideotexString	GeneralString

The following useful types are defined in section three:

GeneralizedTime	EXTERNAL
UTCTime	ObjectDescriptor

The following productions are used in section four:

```
Subtype ::=
        ParentType SubtypeSpec |
        SET SizeConstraint OF Type |
        SEQUENCE SizeConstraint OF Type

ParentType ::= Type

SubtypeSpec ::=
        (SubtypeValueSet SubtypeValueSetList)

SubtypeValueSetList ::=
        "|"
        SubtypeValueSet
        SubtypeValueSetList |
        empty

SubtypeValueSet ::=
        SingleValue |
        ContainedSubtype |
        ValueRange |
        PermittedAlphabet | SizeConstraint | InnerTypeConstraint
        SingleValue ::= Value
        ContainedSubtype ::= INCLUDES Type
        ValueRange ::= LowerEndPoint .. UpperEndPoint
```

```
LowerEndpoint ::= LowerEndValue | LowerEndValue <

UpperEndpoint ::= UpperEndValue | <UpperEndValue

LowerEndValue ::= Value | MIN

UpperEndValue ::= Value | MAX

SizeConstraint ::= SIZE SubtypeSpec

PermittedAlphabet ::= FROM SubtypeSpec
        InnerTypeConstraints ::=

WITH  COMPONENT SingleTypeConstraint |
WITH  COMPONENTS MultipleTypeConstraints

        SingleTypeConstraint ::= SubtypeSpec

MultipleTypeConstraints ::=
        FullSpecification | PartialSpecification

FullSpecification ::= {TypeConstraints}

PartialSpecification ::= {..., TypeConstraints}

TypeConstraints ::=
        NamedConstraint |
        NamedConstraint, TypeConstraints

NamedConstraint ::= identifier Constraint | Constraint
        Constraint ::= ValueConstraint PresenceConstraint
        ValueConstraint ::= SubtypeSpec | empty
        PresenceConstraint ::= PRESENT | ABSENT | empty | OPTIONAL
```

The following additional items are defined in § A.2 for use in the macro notation:

macroreference	"number"
productionreference	"empty"
localtypereference	MACRO
localvaluereference	TYPE
"|"	NOTATION
>	VALUE
astring	value
"string"	type
"identifier"	

The following productions are used in Annex A, with the above items, and items listed at the head of this appendix, as terminal symbols:

```
MacroDefinition ::=
        macroreference
        MACRO
        "::="
        MacroSubstance

MacroSubstance ::=
        BEGIN MacroBody END |
        macroreference |
        Externalmacroreference

MacroBody ::=
        TypeProduction
        ValueProduction
        SupportingProductions

TypeProduction ::=
        TYPE NOTATION
        "::="
        MacroAlternativeList
```

```
ValueProduction ::=
        VALUE NOTATION
        "::="
        MacroAlternativeList

SupportingProductions ::= ProductionList | empty

ProductionList ::= Production | ProductionList Production

Production ::=
        productionreference
        "::="
        MacroAlternativeList

Externalmacroreference ::=
        modulereference . macroreference

MacroAlternativeList ::=
        MacroAlternative |
        MacroAlternativeList "|" MacroAlternative

MacroAlternative ::= SymbolList

SymbolList ::= SymbolElement | SymbolList SymbolElement

SymbolElement ::= SymbolDefn | EmbeddedDefinitions

SymbolDefn ::=
        astring |
        productionreference |
        "string" |
        "identifier" |
        "number" |
        "empty" |
        type |
        type(localtypereference) |
        value(MacroType) |
        value(localvaluereference MacroType) |
        value(VALUE MacroType)

MacroType ::= localtypereference | Type

EmbeddedDefinitions ::= <EmbeddedDefinitionList>

EmbeddedDefinitionList ::=
        EmbeddedDefinition |
        EmbeddedDefinitionList EmbeddedDefinition

EmbeddedDefinition ::=
        LocalTypeassignment |
        LocalValueassignment

LocalTypeassignment ::=
        Localtypereference
        "::="
        MacroType

LocalValueassignment ::=
        localvaluereference
        MacroType
        "::="
        MacroValue

MacroValue ::= Value | localvaluereference
```

SPECIFICATION OF BASIC ENCODING RULES FOR ABSTRACT SYNTAX NOTATION ONE (ASN.1)[1]

(Melbourne, 1988)

The CCITT,

considering

(a) the variety and complexity of information objects conveyed within the application layer;

(b) the need for a high-level notation for specifying such information objects;

(c) the value of isolating and standardizing the rules for encoding such information objects.

unanimously recommends

that the rules for encoding information objects are defined in this Recommendation.

CONTENTS

[1] Recommendation X.209 and ISO 8825 [Information processing systems – Open systems interconnection – Specification of basic encoding rules for abstract Syntax Notation One (ASN.1)] as extended by Addendum 1 to ISO 8825, were developed in close cooperation and are technically aligned.

0 Introduction

Recommendation X.208 (Specification of Abstract Syntax Notation One) specifies a notation for the definition of abstract syntaxes, enabling application layer specifications to define the types of information they need to transfer using the presentation service. It also specifies a notation for the specification of value of a defined type.

This Recommendation defines a set of encoding rules that may be applied to values of types defined using the notation specified in Recommendation X.208. Application of these encoding rules produces a transfer syntax for such values. It is implicit in the specification of these encoding rules that they are also to be used for decoding.

There may be more than one set of encoding rules that can be applied to values of types that are defined using the notation of Recommendation X.208. This Recommendation defines one set of encoding rules, called **basic encoding rules.**

This Recommendation is technically and editorially aligned with ISO 8825 plus Addendum 1 to ISO 8825.

Appendix I gives examples of the application of the encoding rules. It is not part of this Recommendation.

Appendix II summarises the assignment of object identifier values made in this Recommendation and is not part of this Recommendation.

Appendix III is not part of this Recommendation, and gives examples of applying the rules for encoding reals.

1 Scope and field of application

This Recommendation specifies a set of basic encoding rules that may be used to derive the specification of a transfer syntax for values of types defined using the notation specified in Recommendation X.208. These basic encoding rules are also to be applied for decoding such a transfer syntax in order to identify the data values being transferred.

These basic encoding rules are used at the time of communication (by the presentation service provider when required by a presentation context).

2 References

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

[2] Recommendation X.208, *Specification of Abstract Syntax Notation One (ASN.1)* (see also ISO 8824).

[3] Recommendation X.226, *Presentation Protocol Specification for Open Systems Interconnection for CCITT Applications* (see also ISO 8823).

[4] ISO 2022, *Information processing – ISO 7-bit and 8-bit coded character sets – Code extension techniques.*

[5] ISO 2375, *Data processing – Procedure for registration of escape sequences.*

[6] ISO 6093, *Information processing – Representation of numerical values in character strings for information interchange.*

3 Definitions

The definitions of Recommendation X.208 are used in this Recommendation.

3.1 dynamic conformance

A statement of the requirement for an implementation to adhere to the behaviour prescribed by this Recommendation in an instance of communication.

3.2 static conformance

A statement of the requirement for support by an implementation of a valid set of features from among those defined by this Recommendation.

3.3 data value

Information specified as the value of a type; the type and the value are defined using ASN.1.

3.4 encoding (of a data value)

The complete sequence of octets used to represent the data value.

Note – Some CCITT Recommendations use the term "data element" for this sequence of octets, but the term is not used in this Recommendation, as ISO International Standard use it to mean "data value".

3.5 identifier octets

Part of a data value encoding which is used to identify the type of the value.

3.6 length octets

Part of a data value encoding following the identifier octets which is used to determine the end of the encoding.

3.7 end-of-contents octets

Part of a data value encoding, occurring at its end, which is used to determine the end of the encoding.

Note – Not all encodings require end-of-contents octets.

3.8 **contents octets**

That part of a data value encoding which represents a particular value, to distinguish it from other values of the same type.

3.9 **primitive encoding**

A data value encoding in which the contents octets directly represent the value.

3.10 **constructed encoding**

A data value encoding in which the contents octets are the complete encoding of one or more other data values.

3.11 **sender**

An implementation encoding a data value for transfer.

3.12 **receiver**

An implementation decoding the octets produced by a sender, in order to identify the datavalue which was encoded.

4 Abbrevistions and notation

4.1 *Abbreviations*

ASN.1 Abstract Syntax Notation One

4.2 *Notation*

4.2.1 This Recommendation references the notation defined by Recommendation X.208.

4.2.2 This Recommendation specifies the value of each octet in an encoding by use of the terms "most significant bit" and "least significant bit".

Note – Lower layer specifications use the same notation to define the order of bit transmission on a serial line, or the assignment of bits to parallel channels.

4.2.3 For the purposes of this Recommendation, the bits of an octet are numbered from 8 to 1, where bit 8 is the "most significant bit", and bit 1 is the "least significant bit".

5 Conformance

5.1 Dynamic conformance is specified by § 6 to § 24 inclusive.

5.2 Static conformance is specified by those documents which specify the application of these basic encoding rules.

5.3 Alternative encodings are permitted by this Recommendation as a sender's option. Conforming receivers shall support all alternatives.

Note – Examples of such alternative encodings appear in § 6.3.2 b) and Table 2/X.209.

6 General rules for encoding

6.1 *Structure of an encoding*

6.1.1 The encoding of a data value shall consist of four components which shalll appear in the following order:

a) identifier octets (see § 6.2);
b) length octets (see § 6.3);
c) contents octets (see § 6.4);
d) end-of-contents octets (see § 6.5).

6.1.2 The end-of-contents octets shall not be present unless the value of the length octets requires them to be present (see § 6.3).

6.1.3 Figure 1/X.209 illustrates the structure of an encoding (primitive or constructed). Figure 2/X.209 illustrates an alternative constructed encoding.

6.2 *Identifier octets*

6.2.1 The identifier octets shall encode the ASN.1 tag (class and number) of the type of the data value.

6.2.2 For tags with a number ranging from zero to 30 (inclusive), the identifier octets shall comprise a single octets encoded as follows:

a) bits 8 and 7 shall be encoded to represent the clas of the tag as specified in Table 1/X.209;

b) bit 6 shall be a zero or a one according to the rules of § 6.2.5;

c) bit 5 to 1 shall encode the number of the tag as a binary integer with bit 5 as the most significant bit.

6.2.3 Figure 3/X.209 illustrates the form of an identifier octet for a type with a tag whose number is in the range zero to 30 (inclusive).

6.2.4 For tags with a number greater than or equal to 31, the identifier shall comprise a leading octet followed by one or more subsequent octets.

TABLE 1/X.209

Encoding of class of tag

Class	Bit 8	Bit 7
Universal	0	0
Application	0	1
Context-specific	1	0
Private	1	1

6.2.4.1 The leading octet shall be encoded as follows:

a) bits 8 and 7 shall be encoded to represent the class of the tags as listed in Table 1/X.209;

b) bit 6 shall be a zero or a one according to the rule of § 6.2.5;

c) bit 5 to 1 shall be encoded as 11111_2.

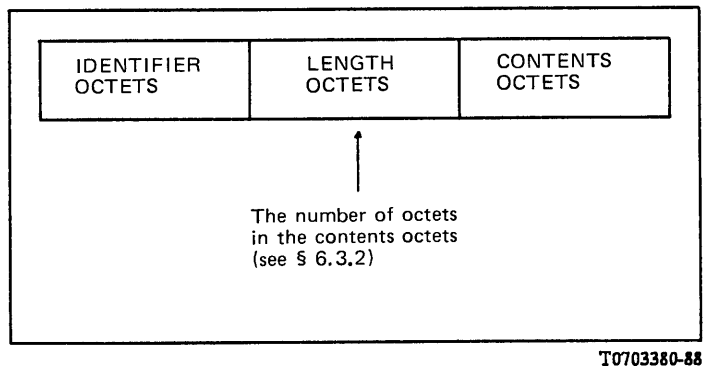


FIGURE 1/X.209

Structure of an encoding

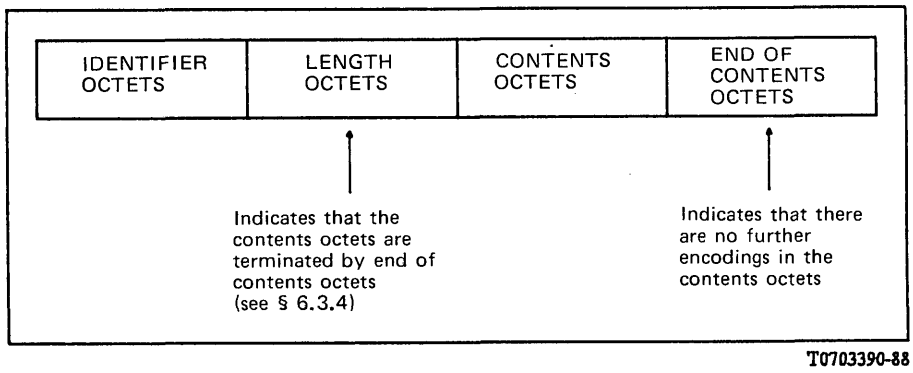


FIGURE 2/X.209

An alternative constructed encoding

6.2.4.2 The subsequent octets shall encode the number of the tag as follows:

a) bit 8 of each octet shall be set to one unless it is the last octet of the identifier octets;

b) bit 7 to 1 of the first subsequent octet, followed by bits 7 to 1 of the second subsequent octet, followed in turn by bits 7 to 1 of each further octet, up to and including the last subsequent octet in the identifier octets shall be the encoding of an unsigned binary integer equal to the tag number, with bit 7 of the first subsequent octet as the most significant bit;

c) bits 7 to 1 of the first subsequent octet shall not all be zero.

6.2.4.3 Figure 4/X.209 illustrates the form of the identifier octets for a type with a tag whose number is greater than 30.

6.2.5 Bit 6 shall be set to zero if the encoding is primitive, and shall be set to one if the encoding is constructed.

Note – Subsequent clauses specify whether the encoding is primitive or constructed for each type.

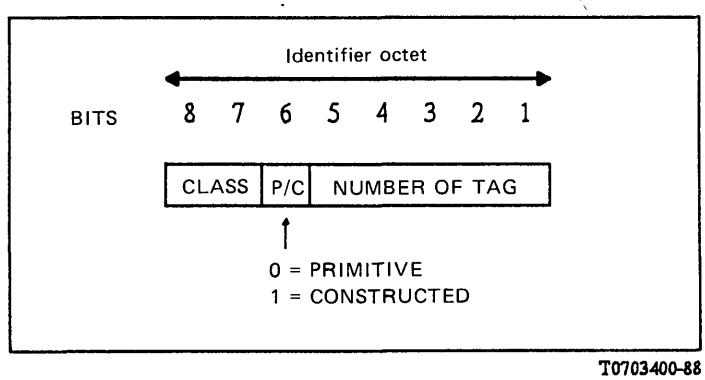


FIGURE 3/X.209

Identifier octet (low tag number)

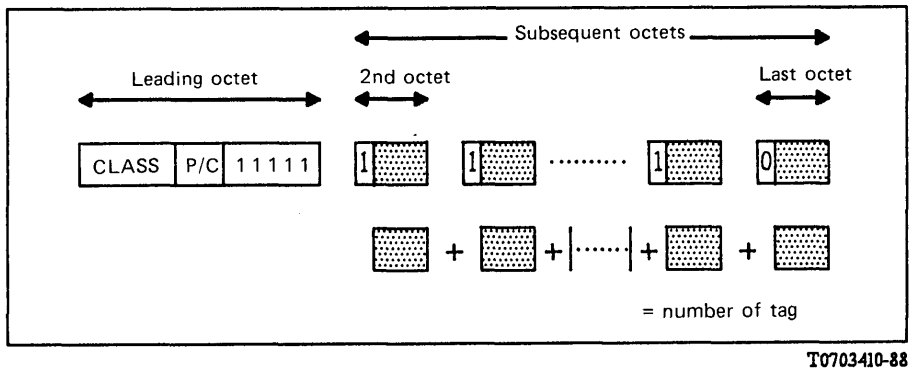


FIGURE 4/X.209

Identifier octets (high tag number)

6.2.6 Recommendation X.208 specifies that the tag of a type defined using the "CHOICE" keyword takes the value of the tag of the type from which the chosen data value is taken.

6.2.7 Recommendation X.208 specifies that the tag of a type defined using "ANY" is indeterminate. The "ANY" type is subsequently defined to be an ASN.1 type, and the complete encoding is then identical to that of a value of the assigned type (including the identifier octets).

6.3 *Length octets*

6.3.1 Two forms of length octets specified. These are

a) the definite form (see § 6.3.3); and

b) the indefinite form (see § 6.3.4).

6.3.2 A sender shall

a) use the definite form (§ 6.3.3) if the encoding is primitive;

b) use either the definite form (§ 6.3.3) or the indefinite form (§ 6.3.4), a sender's option, if the encoding is constructed and all immediately available;

c) use the indefinite form (§ 6.3.4) if the encoding is constructed and is not all immediately available.

6.3.3 For the definite form, the length octets shall consist of one or more octets, and shall represent the number of octets in the contents octets using either the short form (§ 6.3.3.1) or the long form (§ 6.3.3.2) as a sender's option.

Note – The short form can only be used if the number of octets in the contents octets is less than or equal to 127.

6.3.3.1 In the short form, the length octets shall consist of a single octet in which bit 8 is zero and bit 7 to 1 encode the number of octets in the contents octets (which may be zero), as an unsigned binary integer with bit 7 as the most significant bit.

Example:

L = 38 can be encoded as 00100110_2

6.3.3.2 In the long form, the length octets shall consist of an initial octet and one or more subsequent octets. The initial octet shall be encoded as follows:

a) bit 8 shall be one;

b) bits 7 to 1 shall encode the number of subsequent octets in the length octets, as an unsigned binary integer with bit 7 as the most significant bit;

c) the value 11111111_2 shall not be used

Note – This restriction is introduced for possible future extension.

Bits 8 to 1 of the first subsequent octet, followed by bits 8 to 1 of the second subsequent octet, followed in turn by bits 8 to 1 of each further octet up to and including the last subsequent octet, shall be the encoding of an unsigned binary integer equal to the number of octets in the contents octets, with bit 8 of the first subsequent octet as the most significant bit.

Example:

L = 201 can be encoded as: 10000001_2
11001001_2

Note – In the long form, it is a sender's option whether to use more length octets than the minimum necessary.

6.3.4 For the indefinite form, the length octets indicate that the contents are terminated by end-of-contents octets (see § 6.5), and shall consist of a single octet.

6.3.4.1 The single octet shall have bit 8 set to one, and bits 7 to 1 set to zero.

6.3.4.2 If this form of length is used, then end-of-contents octets (see § 6.5) shall be present in the encoding following the contents octets.

6.4 *Contents octets*

The contents octets shall consist of zero, one or more octets, and shall encode the data value as specified in subsequent clauses.

Note – The contents octets depend on the type of the data value; subsequent clauses follow the same sequence as the definition of types in ASN.1.

6.5 *End-of-contents octets*

The end-of-contents octets shall be present if the length is encoded as specified in § 6.3.4, otherwise they shall not be present.

The end-of-contents octets shall consist of two zero octets.

Note – The end-of-contents octets can be considered as the encoding of a value whose tag is universal, whose form is primitive, whose number of the tag is zero, and whose contents is absent, thus:

End-of-contents	Length	Contents
00_{16}	00_{16}	Absent

7 Encoding of a Boolean value

7.1 The encoding of a Boolean value shall be primitive. The contents octets shall consist of a single octet.

7.2 If the Boolean value is

FALSE

the octet shall be zero.

7.2.1 If the Boolean value is

TRUE

the octet shall have any non-zero value, as a sender's option.

Example – If of type BOOLEAN, the value TRUE can be encoded as:

Boolean	Length	Contents
01_{16}	01_{16}	FF_{16}

8 Encoding of an integer value

8.1 The encoding of an integer value shall be primitive. The contents octets shall consist of one or more octets.

8.2 If the contents octets of an integer value encoding consist of more than one octet, then the bits of the first octet and bit 8 of the second octet

a) shall not all be ones; and

b) shall not all be zero.

Note – These rules ensure that an integer value is always encoded in the smallest possible number of octets.

8.3 The contents octets shall be a two's complement binary number equal to the integer value, and consisting of bits 8 to 1 of the first octet, followed by bits 8 to 1 of the second octet, followed by bits 8 to 1 of each octet in turn up to and including the last octet of the contents octets.

Note – The value of a two's complement binary number is derived by numbering the bits in the contents octets, starting with bit 1 of the last octet as bit zero and ending the numbering with bit 8 of the first octet. Each bit is assigned a numerical value of 2^N, where N is its position in the above numbering sequence. The value of the two's complement binary number is obtained by summing the numerical values assigned to each bit for those bits which are set to one, excluding bit 8 of the first octet, and then reducing this value by the numerical value assigned to bit 8 of the first octet if that bit is set to one.

9 Encoding of an enumerated value

9.1 The encoding of an enumerated value shall be that of the integer value with which it is associated.

10 Encoding of a real value

10.1 The encoding of a real value shall be primitive.

10.2 If the real value is the value zero, there shall be no contents octets in the encoding.

10.3 If the real value is non-zero, then the base used for the encoding shall be B', chosen by the sender. If B' is 2, 8 or 16, a binary encoding, specified in § 10.5, shall be used. If B' is 10, a character encoding, specified in § 10.6, shall be used.

Note – The form of storage, generation, or processing by senders and receivers, and the form used in the ASN.1 value notation are all independent of the base used for transfer.

10.4 Bit 8 of the first contents octet shall be set as follows:

a) if bit 8 = 1, then the binary encoding specified in § 10.5 applies;

b) if bit 8 = 0 and bit 7 = 0, then the decimal encoding specified in § 10.6 applies;

c) if bit 8 = 0 and bit 7 = 1, then a "SpecialRealValue" (see Recommendation X.208) is encoded as specified in § 10.7.

10.5 When binary encoding is used (bit 8 = 1), then if the mantissa, M is non-zero, it shall be represented by a sign S, a non-negative integer value N and a binary scaling factor F, such that

$$M = S \times N \times 2^F, \; 0 \leqslant F < 4, \; S = +1 \text{ or } -1$$

Note – This freedom to choose F is provided to enable easier generation of the transfer format by eliminating the need to align the implied decimal point of the mantissa with an octet boundary (see Appendix III). The existence of F does not noticeably complicate the task of receivers.

10.5.1 Bit 7 of the first contents octets shall be 1 if S is -1 and 0 otherwise.

10.5.2 Bits 6 to 5 of the first contents octets shall encode the value of the base B' as follows:

Bits 6 to 5	Base
00	base 2
01	base 8
10	base 16
11	Reserved for future versions of this Recommendation

10.5.3 Bits 4 to 3 of the first contents octet shall encode the value of the binary scaling factor F as an unsigned binary integer.

10.5.4 Bits 2 to 1 of the first contents octet shall encode the format of the exponent as follows:

a) if bits 2 to 1 are 00, then the second contents octet encodes the value of the exponent as a two's complement binary number;

b) if bits 2 to 1 are 01, then the second and third contents octets encode the value of the exponents as a two's complement binary number;

c) if bits 2 to 1 are 10, then the second, third and fourth contents octets encode the value of the exponent as a two's complement binary number;

d) if bits 2 to 1 are 11, then the second contents octet encodes the number of octets, X say, (as an unsigned binary number) used to encode the value of the exponent, and the third up to the (X plus 3)th (inclusive) contents octets encode the value of the exponent as a two's complement binary number; the value of X shall be at least one; the first nine bits of the transmitted exponent shall not be all zeros or all ones.

10.5.5 The remaining contents octets encode the value of the integer N (see § 10.5) as an unsigned binary number.

Note 1 – This encoding does not specify a "normalized" representation, there being a number of possible representations of each value (except zero). This variation is a senders option, and can be used as a broad indication of precision.

Note 2 – This representation of real numbers is very different from the formats normally used in floating point hardware, but has been designed to be easily converted to and from such formats (see Appendix III).

10.6 When decimal encoding is used (bits 8 to 7 = 00), all the contents octets following the first contents octet form a field, as the term is used in ISO 6093, of a length chosen by the sender, and encoded according to ISO 6093. The choice of ISO 6093 number representation is specified by bits 6 to 1 of the first contents octet as follows:

Bits 6 to 1	Number representation
00 0001	ISO 6093 NR1 form
00 0010	ISO 6093 NR2 form
00 0011	ISO 6093 NR3 form

The remaining values of bits 6 to 1 are reserved for future versions of this Recommendation.

Note 1 – The Recommendation in ISO 6093 concerning the user of at least one digit to the left of the decimal mark are also recommended in this Recommendation, but are not mandatory.

Note 2 – There shall be no use of scaling factors specified in accompanying documentation (see ISO 6093).

Note 3 – Use of the normalised form (see ISO 6093) is a senders option, and has no significance.

10.7 When "SpecialRealValues" are to be encoded (bits 8 to 7 = 01), there shall be only one contents octet, with values as follows:

01000000	Value is PLUS-INFINITY
01000001	Value is MINUS-INFINITY

All other values having bit 8 and 7 equal to 0 and 1 respectively are reserved for future versions of this Recommendation.

11 Encoding of a bitstring value

11.1 The encoding of a bitstring value shall be either primitive or constructed at the option of the sender.

Note – Where it is necessary to transfer part of a bit string before the entire bitstring is available, the constructed encoding is used.

11.2 The contents octets for the primitive encoding shall contain an initial octet followed by zero, one or more subsequent octets.

11.2.1 The bits in the bitstring, commencing with the first bit and proceeding to the trailing bit, shall be placed in bits 8 to 1 of the first subsequent octet, followed by bits 8 to 1 of the second subsequent octet, followed by bits 8 to 1 of each octet in turn, followed by as many bits as are needed of the final subsequent octet, commencing with bit 8.

Note – The notation "first bit" and "trailing bit" is specified in Recommendation X.208.

11.2.2 The initial octet shall encode, as an unsigned binary integer with bit 1 as the least significant bit, the number of unused bits in the final subsequent octet. The number shall be in the range zero to seven.

11.2.3 If the bitstring is empty, there shall be no subsequent octets, and the initial octet shall be zero.

11.3 The contents octets for the constructed encoding shall consist of the complete encoding of zero, one or more data values.

Note – Each such encoding includes identifier, length, and contents octets, and may include end-of-contents octets if it is constructed.

11.3.1 Each data value encoding in the contents octets shall be the encoding of a value of type BIT STRING.

Note – In particular, the tags in the contents octets are always universal class, number 3.

11.3.2 The bits in the bitstring value being encoded, commencing with the first bit and proceeding to the trailing bit, shall be placed in the first bit up to the trailing bit of the first data value encoded in the contents octets, followed by the first bit up to the trailing bit of the second data value encoded in the contents octets, followed by the first bit up to the trailing bit of each data value in turn, followed by the first bit up to the trailing bit of the last data value encoded in the contents octets.

11.3.3 Each data value encoded in the contents octets, with the exception of the last, shall consist of a number of bits which is a multiple of eight.

Note – A data value encoded in the contents octets may be a zero-length bitstring.

11.3.4 Where a constructed encoding is used, there shall be no significance placed on the boundary between the data values encoded in the contents octets.

11.3.5 The encoding of each data value encoded in the contents octets may be primitive or constructed.

Note – It is usually primitive.

Example – If of type BIT STRING, the value '0A3B5F291CD'H can be encoded as shown below. In this example, the Bit String is represented as a primitive:

BitString	Length	Contents
03_{16}	07_{16}	$040A3B5F291CD0_{16}$

The value shown above can also be encoded as shown below. In this example, the Bit String is represented as a constructor:

BitString	*Length*	*Contents*		
23_{16}	80_{16}			
		BitString	*Length*	*Contents*
		03_{16}	03_{16}	$000A3B_{16}$
		BitString	*Length*	*Contents*
		03_{16}	05_{16}	$045F291CD0_{16}$
		EOC	*Length*	
		00_{16}	00_{16}	

12 Encoding of an octetstring value

12.1 The encoding of an octetstring value shall be either primitive or constructed at the option of the sender.

Note – Where it is necessary to transfer part of an octet string before the entire octetstring is available, the constructed encoding is used.

12.2 The primitive encoding contains zero, one or more contents octets equal in value to the octets in the data value, in the order they appear in the data value, and with the most significant bit of an octet of the data value aligned with the most significant bit of an octet of the contents octets.

12.3 The contents octets for the constructed encoding shall consist of the complete encoding of zero, one or more data values.

Note – Each such encoding includes identifier, length, and contents octets, and may include end-of-contents octets if it is constructed.

12.3.1 Each data value encoding in the contents octets shall be the encoding of a value of type octetstring.

Note – In particular, the tags in the contents octets are always universal class, number 4.

12.3.2 The octets in the octetstring value being encoded, commencing with the first octet and proceeding to the trailing octet, shall be placed in the first up to the trailing octet of the first data value encoded in the contents octets, followed by the first up to the trailing octet of the second data value encoded in the contents octets, followed by the first up to the trailing octet of each data value in turn, followed by the first up to the trailing octet of the last data value encoded in the contents octets.

Note – A data value encoded in the contents octets may be a zero length octet string.

12.3.3 Where a constructed encoding is used, there shall be no significance placed on the boundary between the data values encoded in the contents octets.

12.3.4 The encoding of each data value encoded in the contents octets may be primitive or constructed.

Note – It is usually primitive.

13 Encoding of a null value

13.1 The encoding of a null value shall be primitive.

13.2 The contents octets shall not contain any octets.

Note – The length octet is zero.

Example – If of type NULL, the NULL can be encoded as:

Null	*Length*
05_{16}	00_{16}

14 Encoding of a sequence value

14.1 The encoding of a sequence value shall be constructed.

14.2 The contents octets shall consist of the complete encoding of one data value from each of the types listed in the ASN.1 definition of the sequence type, in the order of their appearance in the definition, unless the type was referenced with the keyword "OPTIONAL" or the keyword "DEFAULT".

14.3 The encoding of a data value may, but need not, be present for a type which was referenced with the keyword "OPTIONAL" or the keyword "DEFAULT". If present, it shall appear in the encoding at the point corresponding to the appearance of the type in the ASN.1 definition.

Example – If of type

SEQUENCE {name IA5String, ok BOOLEAN}

the value

{name "Smith", ok TRUE}

can be encoded as:

Sequence	*Length*	*Contents*		
30_{16}	$0A_{16}$			
		IA5String	*Length*	*Contents*
		16_{16}	05_{16}	"Smith"
		Boolean	*Length*	*Contents*
		01_{16}	01_{16}	FF_{16}

15 Encoding of a sequence-of value

15.1 The encoding of a sequence-of value shall be constructed.

15.2 The contents octets shall consist of zero, one or more complete encodings of data values from the type listed in the ASN.1 definition.

15.3 The order of the encodings of the data values shall be the same as the order of the data values in the sequence-of value to be encoded.

16 Encoding of a set value

16.1 The encoding of a set value shall be constructed.

16.2 The contents octets shall consist of the complete encoding of a data value from each of the types listed in the ASN.1 definition of the set type, in an order chosen by the sender, unless the type was referenced with the keyword "OPTIONAL" or the keyword "DEFAULT".

16.3 The encoding of a data value may, but need not, be present for a type which was referenced with the keyword "OPTIONAL" or the keyword "DEFAULT".

Note – The order of data values in a set value is not significant, and places no constraints on the order during transfer.

17 Encoding of a set-of value

17.1 The encoding of a set-of value shall be constructed.

17.2 The text of § 15.2 applies.

17.3 The order of data values need not be preserved by the encoding and subsequent decoding.

18 Encoding of a choice value

The encoding of a choice value shall be the same as the encoding of a value of the chosen type.

Note 1 – The encoding may be primitive or constructed depending on the chosen type.

Note 2 – The tag used in the identifier octets is the tag of the chosen type, as specified in the ASN.1 definition of the choice type.

19 Encoding of a selection value

The encoding of a selection value shall be the same as the encoding of a value of the selected type.

Note – The encoding may be primitive or constructed depending on the selected type.

20 Encoding of a tagged value

20.1 The encoding of a tagged value shall be derived from the complete encoding of the corresponding data value of the type appearing in the "TaggedType" notation (called the base encoding) as specified in § 20.2 and § 20.3.

20.2 If the "IMPLICIT" keyword was not used in the definition of the type, the encoding shall be constructed and the contents octets shall be the complete base encoding.

20.3 If the "IMPLICIT" keyword was used in the definition of the type, then

a) the encoding shall be constructed if the base encoding is constructed, and shall be primitive otherwise; and

b) the contents octets shall be the same as the contents octets of the base encoding.

Example – With ASN.1 type definitions of

```
Type1 ::= VisibleString
Type2 ::= [APPLICATION 3] IMPLICIT Type1
Type3 ::= [2] Type2
Type4 ::= [APPLICATION 7] IMPLICIT Type3
Type5 ::= [2] IMPLICIT Type2
```

a value of

"Jones"

is encoded as follows:

For Type1:

VisibleString	*Length*	*Contents*
$1A_{16}$	05_{16}	$4A6F6E6573_{16}$

For Type2:

[Application 3]	*Length*	*Contents*
43_{16}	05_{16}	$4A6F6E6573_{16}$

For Type3:

[2]	*Length*	*Contents*
$A2_{16}$	07_{16}	
[Application 3]	*Length*	*Contents*
43_{16}	05_{16}	$4A6F6E6573_{16}$

For Type4:

[Application 7]	*Length*	*Contents*
67_{16}	07_{16}	
[Application 3]	*Length*	*Contents*
43_{16}	05_{16}	$4A6F6E6573_{16}$

For Type5:

[2]	*Length*	*Contents*
82_{16}	05_{16}	$4A6F6E6573_{16}$

21 Encoding of a value of the ANY type

The encoding of an ANY type shall be the complete encoding specified in this Recommendation for the type of the value of the ANY type.

22 Encoding of an object identifier value

22.1 The encoding of an object identifier value shall be primitive.

22.2 The contents octets shall be an (ordered) list of encoding of subidentifiers (see § 22.3 and § 22.4) concatenated together.

Each subidentifier is represented as a series of (one or more) octets. Bit 8 of each octet indicates whether it is the last in the series: bit 8 of the last octet is zero; bit 8 of each preceding octet is one. Bits 7-1 of the octets in the series collectively encoded the subidentifier. Conceptually, these groups of bits are concatenated to form an unsigned binary number whose most significant bit is bit 7 of the first octet and whose least significant bit is bit 1 of the last octet. The subidentifier shall be encoded in the fewest possible octets, that is, the leading octet of the subidentifier shall not have the value 80 (hexadecimal).

22.3 The number of subidentifiers (N) shall be one less than the number of object identifier components in the object identifier value being encoded.

22.4 The numerical value of the first subidentifier is derived from the values of the first two object identifier components in the object identifier value being encoded, using the formula

$$(X * 40) + Y$$

where X is the value of the first object identifier component and Y is the value of the second object identifier component.

Note – This packing of the first two object identifier components recognises that only three values are allocated from the root node, and at most 39 subsequent values from nodes reached by X = 0 and X = 1.

22.5 The numerical value of the i'th subidentifier, ($2 \leqslant i \leqslant N$) is that of the (i + 1)'th object identifier component.

Example – An OBJECT IDENTIFIER value of

{joint-iso-ccitt 100 3}

which is the same as

{2 100 3}

has a first subidentifier of 180 and a second subidentifier of 3. The resulting encoding is

OBJECT IDENTIFIER	*Length*	*Contents*
06_{16}	03_{16}	813403_{16}

23 Encoding for values of the character string types

23.1 The data value consists of a string of characters from the character set specified in the ASN.1 type definition.

23.2 Each data value shall be encoded independently of other data values of the same type.

23.3 Each character string type shall be encoded as if it had been declared

[UNIVERSAL x] IMPLICIT OCTET STRING

where x is the number of the universal class tag assigned to the character string type in Recommendation X.208. The value of the octet string is specified in §§ 23.4 and 23.5.

23.4 Where a character string type is specified in Recommendation X.208 by direct reference to an enumerating table (NumericString and PrintableString), the value of the octet string shall be that specified in § 23.5 for a VisibleString type with the same character string value.

23.5 The octet string shall contain the octets specified in ISO 2022 for encodings in an 8-bit environment, using the escape sequence and character codings registered in accordance with ISO 2375.

23.5.1 An escape sequence shall not be used unless it is one of those specified by one of the registration numbers used to define the character string type in Recommendation X.208.

TABLE 2/X.209

Use of escape sequences

Type	Assumed G0 (Registration number)	Assumed C0 & C1 (Registration number)	Assumed escape sequence(s) and locking shift (where applicable)	Explicit escape sequences allowed?
NumericString	2	None	ESC 2/8 4/0 LS0	NO
PrintableString	2	None	ESC 2/8 4/0 LS0	NO
TeletexString (T61String)	102	106(C0) 107(C1)	ESC 2/8 7/5 LS0 ESC 2/1 4/5 ESC 2/2 4/8	YES
VideotexString	102	1(C0) 73(C1)	ESC 2/8 7/5 LS0 ESC 2/1 4/0 ESC 2/2 4/1	YES
VisibleString (ISO646String)	2	None	ESC 2/8 4/0 LS0	NO
IA5String	2	1(C0)	ESC 2/8 4/0 LS0 ESC 2/1 4/0	NO
GraphicString	2	None	ESC 2/8 4/0 LS0	YES
GeneralString	2	1(C0)	ESC 2/8 4/0 LS0 ESC 2/1	YES

Note – Many of the commonly used characters (for example, A to Z) appear in a number of character repertoires with individual registration numbers and escape sequences. Where ASN.1 types allow escape sequences, a number of encodings may be possible for a particular character string (see also § 5.3).

23.5.2 At the start of each string, certain registration numbers shall be assumed to be designated as G0 and/or C0 and/or C1, and invoked (using the terminology of ISO 2022). These are specified for each type in Table 2/X.209, together with the assumed escape sequence they imply.

23.5.3 Certain character string types shall not contain explicit escape sequences in their encodings; in all other cases, any escape allowed by § 23.5.1 can appear at any time, including at the start of the encoding. Table 2/X.209 lists the types for which explicit escape sequences are allowed.

23.5.4 Announcers shall not be used unless explicitly permitted by the user of ASN.1.

Note – The choice of ASN.1 type provides a limited form of announcer functionality. Specific application protocols may choose to carry announcers in other protocol elements, or to specify in detail the manner of use of announcers.

Example – With the ASN.1 type definition

Name ::= VisibleString

a value

"Jones"

can be encoded (primitive form) as

VisibleString	*Length*	*Contents*
$1A_{16}$	05_{16}	$4A6F6E6573_{16}$

or (constructor form, definite length), as:

VisibleString	*Length*	*Contents*
$3A_{16}$	09_{16}	
OctetString	*Length*	*Contents*
04_{16}	03_{16}	$4A6F6E_{16}$
OctetString	*Length*	*Contents*
04_{16}	02_{16}	6573_{16}

or (constructor form, indefinite length), as:

VisibleString	*Length*	*Contents*
$3A_{16}$	80_{16}	
OctetString	*Length*	*Contents*
04_{16}	03_{16}	$4A6F6E_{16}$
OctetString	*Length*	*Contents*
04_{16}	02_{16}	6573_{16}
EOC	*Length*	
00_{16}	00_{16}	

The above example illustrates three of the (many) possible forms available as a sender's option. Receivers are required to handle all permitted forms (see § 5.3).

24 Encoding for values of the ASN.1 useful types

A definition of these types using ASN.1 is provided in Recommendation X.208. The encoding shall be that obtained by applying the rules specified in this Recommendation to that type definition.

25 Use in transfer syntax definition

25.1 The encoding rules specified in this Recommendation can be referenced and applied whenever there is a need to specify an unambiguous, undivided and self-delimiting octet string representation for all of the values of a single ASN.1 type.

Note – All such octet strings are unambiguous within the scope of the single ASN.1 type. They would not necessarily be unambiguous if mixed with encodings of a different ASN.1 type.

25.2 The object identifier and object descriptor values

{joint-iso-ccitt asn1 (1) basic-encoding (1)}

and

"Basic Encoding of a single ASN.1 type"

are assigned to identify and describe the encoding rules specified in this Recommendation.

25.3 Where an application specification defines an abstract syntax as a set of presentation data values, each of which is a value of some specifically named ASN.1 type, usually (but not necessarily) a choice type, then the object identifier value specified in § 25.2 may be used with the abstract syntax name to identify that transfer syntax which results from the application of the encoding rules specified in this Recommendation to the specifically named ASN.1 type used in defining the abstract syntax.

Note – In particular, this identification of the encoding rules can appear in the "transfer syntax name" field of the presentation protocol (Recommendation X.226).

25.4 The name specified in § 25.2 shall not be used with an abstract syntax name to identify a transfer syntax if the conditions of § 25.3 for the definition of the abstract syntax are not met.

APPENDIX I

(to Recommendation X.209)

Example of encodings

This appendix illustrates the basic encoding rules specified in this Recommendation by showing the representation in octets of a (hypothetical) personnel record which is defined using ASN.1.

I.1 *ASN.1 description of the record structure*

The structure of the hypothetical personnel record is formally described below using ASN.1 specified in Recommendation X.208 for defining types.

```
PersonnelRecord ::= [APPLICATION 0] IMPLICIT SET
{                           Name,
 title                      [0] VisibleString,
 number                     EmployeeNumber,
 dateOfHire                 [1] Date,
 nameOfSpouse               [2] Name,
 children                   [3] IMPLICIT
                            SEQUENCE OF ChildInformation
                            DEFAULT {}}

ChildInformation ::= SET
{                           Name,
 dateOfBirth                [0] Date}

Name ::= [APPLICATION 1] IMPLICIT SEQUENCE
{givenName                  VisibleString,
 initial                    VisibleString,
 familyName                 VisibleString}

EmployeeNumber ::= [APPLICATION 2] IMPLICIT INTEGER

Date ::= [APPLICATION 3] IMPLICIT VisibleString
  -- YYYYMMDD
```

I.2 *ASN.1 description of a record value*

The value of John Smith's personnel record is formally described below using ASN.1.

```
{                           {givenName "John",initial "P",familyName "Smith"},
 title                      "Director",
 number                     51,
 dateOfHire                 "19710917",
 nameOfSpouse               {givenName "Mary",initial "T",familyName "Smith"},
 children
   {{{givenName "Ralph",initial "T",familyName "Smith"},
         dateOfBirth "19571111"},
         {{givenName "Susan",initial "B",familyName "Jones"},
           dateOfBirth "19590717"}}}
```

I.3 *Representation of this record value*

The representation in octets of the record value given above (after applying the basic encoding rules defined in this Recommendation) is shown below. The values of identifiers, lengths, and the contents of integers are shown in hexadecimal, two hexadecimal digits per octet. The values of the contents of character strings are shown as text, one character per octet.

Personnel Record 60	*Length* 8185	*Contents*								
		Name 61	*Length* 10	*Contents*						
				Visible-String 1A	*Length* 04	*Contents* "John"				
				Visible-String 1A	*Length* 01	*Contents* "P"				
				Visible-String 1A	*Length* 05	*Contents* "Smith"				
		Title A0	*Length* 0A	*Contents*						
				Visible-String 1A	*Length* 08	*Contents* "Director"				
		Employee Number 42	*Length* 01	*Contents* 33						
		Date of Hire A1	*Length* 0A	*Contents*						
				Date 43	*Length* 08	*Contents* "19710917"				
		Name of Spouse A2	*Length* 12	*Contents*						
				Name 61	*Length* 10	*Contents*				
						Visible-String 1A	*Length* 04	*Contents* "Mary"		
						Visible-String 1A	*Length* 01	*Contents* "T"		
						Visible-String 1A	*Length* 05	*Contents* "Smith"		
		[3] A3	*Length* 42	*Contents*						
				Set 31	*Length* 1F	*Contents*				
						Name 61	*Length* 11	*Contents*		
								Visible-String 1A	*Length* 05	*Contents* "Ralph"
								Visible-String 1A	*Length* 01	*Contents* "T"
								Visible-String 1A	*Length* 05	*Contents* "Smith"
						Date of Birth A0	*Length* 0A	*Contents*		
								Date 43	*Length* 08	*Contents* "19571111"
				Set 31	*Length* 1F	*Contents*				
						Name 61	*Length* 11	*Contents*		
								Visible-String 16	*Length* 05	*Contents* "Susan"
								Visible-String 16	*Length* 01	*Contents* "B"
								Visible-String 1	*Length* 05	*Contents* "Jones"
						Date of Birth A0	*Length* 0A	*Contents*		
								Date 43	*Length* 08	*Contents* "19590717"

APPENDIX II

(to Recommendation X.209)

Assignment of object identifier values

The following values are assigned in this Recommendation:

Clause	**Object Identifier Value** **Object Descriptor Value**
25.2	{joint-iso-ccitt asn1 (1) basic-encoding (1)} "Basic Encoding of a single ASN.1 type"

APPENDIX III

(to Recommendation X.209)

Illustration of real value encoding

III.1 A sender will normally examine his own hardware floating point representation to determine the (value-independent) algorithms to be used to transfer values between this floating-point representation and the length and contents octets of the encoding of an ASN.1 real value. This Appendix illustrates the steps which would be taken in such a process by using the (artificial) hardware floating point representation of the mantissa shown in Figure III-1/X.209.

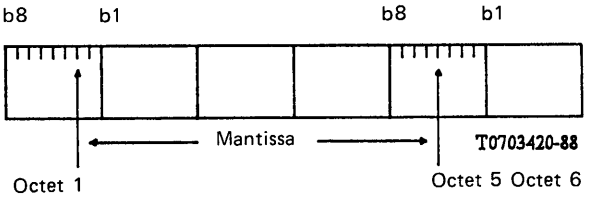


FIGURE III -1/X.209

Hypothetical hardware representation

It is assumed that the exponent can easily be obtained from the floating point hardware as an integer value E.

III.2 The contents octets which need to be generated for sending a non-zero value (as specified in the body of this Recommendation) are:

1 S bb ff ee Octets for E Octets for N

where S (the mantissa sign) is dependent on the value to be converted, bb is a fixed value (say 10) to represent the base (in this case let us assume base 16), ff is the fixed F vaue calculated as described in § III.3, and ee is a fixed length of exponent value calculated as described in § III.4 (this Appendix does not treat the case where E needs to exceed three octets).

III.3 The algorithm will transmit octets 1 to 5 of the hardware representation as the value of N, after forcing bits 8 to 3 of octet 1 and bits 4 to 1 of octet 5 to zero. The implied decimal point is assumed to be positioned between bits 2 and 1 of octet in the hardware representation which delivers the value of E. Its implied position can be shifted to the nearest point after the end of octet 6 by reducing the value of E before transmission. In our example system we can shift by four bits for every exponent decrement (because we are assuming base 16), so a decrement of 9 will position the implied point between bits 6 and 5 of octet 6. Thus the value of M is N multiplied by 2^3 to position the point correctly in M. (The implied position N, the octets transferred, is after bit 1 of octet 5.) Thus we have the crucial parameters:

F = 3 (so ff is 11)
exponent decrement = 9

III.4 The length needed for the exponent is now calculated by working out the maximum number of octets needed to represent the values

E_{min} – excess – exponent decrement
E_{max} – excess – exponent decrement

where E_{min} and E_{max} are minimum and maximum integer values of the exponent representation, excess is any value which needs subtracting to produce the true exponent value, and the exponent decrement is as calculated in § III.3. Let us assume this gives a length of 3 octets. Then ee is 10. Let us also assume excess is zero.

III.5 The transmission algorithm is now:

a) test for zero, and if so, transmit an ASN.1 length of zero (no contents octets) and end the algorithm;

b) test and remember the mantissa sign, and negate the mantissa if negative;

c) transmit an ASN.1 length of 9, then

11101110 if negative

or

10101110 if positive

d) produce and transmit the 3 octet exponent with value

E – 9

e) zero bits 8 to 3 of octet 1 and bits 4 to 1 of octet 5, then transmit the 5 octet mantissa.

III.6 The receiving algorithm has to be prepared to handle any ASN.1 format, but here the floating point unit can be directly used. We proceed as follows:

a) check octet 1 of the contents; if it is 1x101110 we have a transmission compatible with ours, and can simply reverse the sending algorithm;

b) otherwise, for character encoding invoke standard character decimal to floating point conversion software, and deal with a "SpecialRealValue" according to the application semantics (perhaps setting the largest and smallest number the hardware floating point can handle);

c) for a binary transmission, put N into the floating point unit, losing octets at the least significant end if necessary, multiply by 2^F, and by B^E, then negate if necessary. Implementors may find optimisation possible in special cases, but may find (apart from the optimisation relating to transmissions from a compatable machine) that testing for them loses more than they gain.

III.7 The above algorithms are illustrative only. Implementors will, of course, determine their own best strategies.

PAGE INTENTIONALLY LEFT BLANK

PAGE LAISSEE EN BLANC INTENTIONNELLEMENT

SECTION 2

SERVICE DEFINITIONS

Recommendation X.210

OPEN SYSTEMS INTERCONNECTION LAYER SERVICE DEFINITION CONVENTIONS[1)]

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

The CCITT,

considering

(a) that Recommendation X.200 provides a Reference Model which separates the functions of inter-process communications into seven layers;

(b) that the layer services developed for these seven layers of function will generally result in separate Recommendations;

(c) that there is a need for the resulting Recommendations to be a coherent set,

unanimously recommends

that the conventions and terminology defined in this Recommendation be used for Layer Service definitions of Open Systems Interconnection.

CONTENTS

1 Scope and field of application

This Recommendation establishes definitions of terms and conventions for reference by other Recommendations which define the services provided by layers 1 to 6 of the Reference Model of Open Systems Interconnection for CCITT Applications (Recommendation X.200). These conventions are also applicable in specifying other telecommunications and data processing services and capabilities which utilize the layered model concepts and principles prescribed by the Reference Model. In particular it is concerned with conventions relating to just two communicating systems at any one time, and connection-oriented services.

[1)] Recommendation X.210 and ISO TR 8509, [Information processing systems – Open Systems Interconnection – Service conventions] were developed in close collaboration and are technically aligned, except for the differences noted in Appendix II.

2 References

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

3 Definitions

3.1 This Recommendation builds on the concepts developed in Recommendation X.200 and makes use of the following terms defined in that Recommendation:

a) (N) layer;

b) (N) service;

c) (N) entity;

d) (N) service-access-point;

e) (N) service-access-point-address.

Note – The term "service-access-point" is used when describing the relationship between primitives associated with a single connection. Further study is required to include the concept of connection endpoints in this description.

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

3.2.1 service-user

An abstract representation of the totality of those entities in a single system that make use of a service through a single access point.

3.2.2 service-provider

An abstract machine which models the behaviour of the totality of the entities providing the service, as viewed by the user.

3.2.3 service-primitive; primitive

An abstract, implementation independent interaction between a service-user and the service-provider.

3.2.4 request (primitive)

A primitive issued by a service-user to invoke some procedure.

3.2.5 indication (primitive)

A primitive issued by a service-provider either:

i) to invoke some procedure; or

ii) to indicate that a procedure has been invoked by the service-user at the peer service-access-point.

3.2.6 response (primitive)

A primitive issued by a service-user to complete, at a particular service-access-point, some procedure previously invoked by an indication at that service-access-point.

3.2.7 confirm (primitive)

A primitive issued by a service-provider to complete, at a particular service-access-point, some procedure previously invoked by a request at that service-access-point.

Note – Confirms and responses can be positive or negative as appropriate to the circumstances.

3.2.8 **(N)-mandatory-service**

A service which must be provided in the (N)-service.

3.2.9 **(N)-provider-optional-service**

A service which may or may not be provided in the (N)-service.

3.2.10 **(N)-user-optional-service**

A service which will only be provided if the (N)-service-user requests it, and it is available in the (N)-service.

3.2.11 **unconfirmed-service**

A service which does not result in an explicit confirmation.

3.2.12 **confirmed-service**

A service which results in an explicit confirmation from the service-provider. There is not necessarily any relationship to a response from the peer service-user.

3.2.13 **provider-intitiated-service**

A service which is generated by the service-provider.

4 Model for layer services

A layer service is defined in terms of an abstract model having the following elements:

a) (N)-service-users;

b) (N)-service-provider.

For the lifetime of a particular connection each service-user gains access to the service-provider as indicated in Figure 1/X.210.

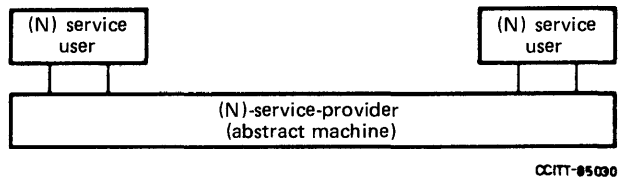
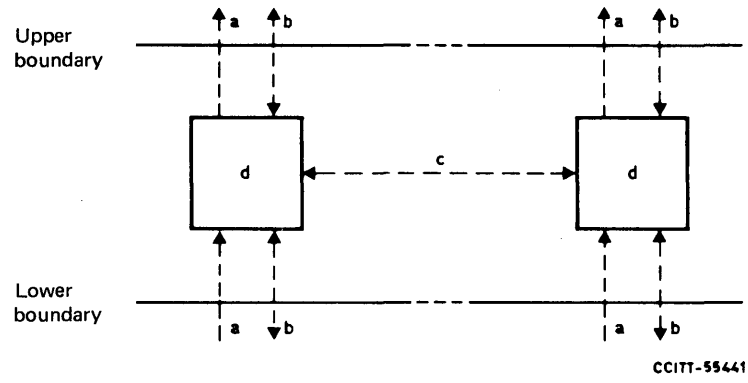


FIGURE 1/X.210

Layer Service Model

Each service-user interacts with the service-provider by issuing or receiving service-primitives. The layer service defines relations between interactions at one service-access-point and consequential interactions at service-access-points used by service-users in order to communicate.

The relationship among the terms service, boundary, service primitive, peer protocol, and peer entities are illustrated in Figure 2/X.210.



a = service
b = service primitives
c = peer protocol
d = peer entities

FIGURE 2/X.210

Relationship among Terms

5 Service primitives

Note – The detailed properties of service primitives are for further study.

5.1 *General*

The use of primitives does not preclude any specific implementation of a service in terms of interface primitives. The following comments apply to this definition technique based on service primitives:

a) service primitives are conceptual, and need not be either directly related to protocol elements, or seen as macro calls of an access method to the layer service;

b) there are other equivalent sets of service primitives which can describe the same layer service;

c) only service primitives which correspond to some element of the layer service involving two service-users need to be considered. The primitives which are only related to local conventions between the service-user and service-provider do not relate to this description technique. For example, strictly local functions could be provided in some implementations. As they do not involve both users, such functions are not visible outside the local system.

5.2 *Categories of service*

The following types of service are identified:

a) mandatory-service (see § 3.2.8);

b) provider-optional-service (see § 3.2.9);

c) user-optional-service (see § 3.2.10).

A user optional service may be either a mandatory service or a provider optional service.

5.3 *Types of service primitives*

Four types of service primitives are identified:

a) request primitive (see § 3.2.4);

b) indication primitive (see § 3.2.5);

c) response primitive (see § 3.2.6);

d) confirm primitive (see § 3.2.7).

5.4 *Properties of primitives*

An individual service primitive is a logically separate interaction which cannot be interrupted by another interaction. A service primitive has a direction which is either:

a) from a service-user to the service-provider;

b) from the service-provider to a service-user.

One or more parameters may be associated with a service primitive and each of these parameters has a defined range of values. Parameter values associated with a service primitive are passed in the direction of the service primitive.

5.5 *Names of primitives*

The name of each service primitive contains three elements:

a) an initial (or initials) which specifies the layer (see § I.1);

b) a name which specifies the service-name (see § I.2);

c) a name which specifies the type of primitive (see § I.3).

6 Conventions for time-sequence diagrams

Time-sequence diagrams are used to illustrate how sequences of interactions are related in time.

Time-sequence diagrams (see Figure 3/X.210) indicate:

a) the sequence of events at each user/provider interface;

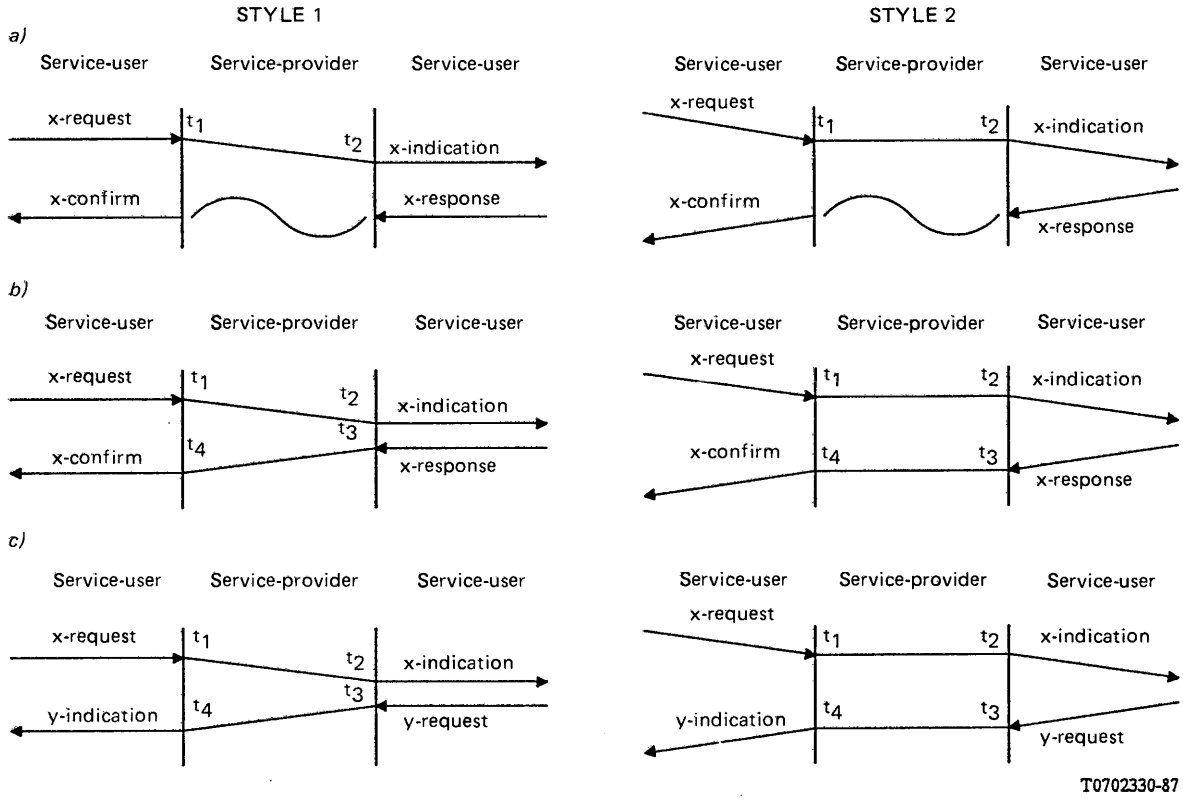
b) where appropriate, the sequence of events between peer users.

Each diagram is partitioned by two vertical lines into three fields. The central field represents the service-provider and the two side fields represent the two service-users. The lines represent the service-access-points between the service-users and the service-provider.

Sequences of events at each service-access-point are positioned along lines representing the passage of time, increasing downwards. Arrows, placed in the areas representing the service-user, indicate the direction of propagation of primitives (i.e., *to* or *from* the service-user) and may include implicit flow control between the service-user and service-provider.

Necessary sequence relations between the two interaction points are emphasized by a solid line between the time lines (e.g., in Figure 3a/X.210 the request primitive from one service-user to the service-provider at time t_1 is necessarily followed by the indication primitive to the peer service-user at time t_2). In the absence of this solid line, there is no specific relationship between the delivery of confirmation and indication. The absence of relationship is indicated either by leaving the central field blank or, for clarity, by use of a tilde (~).

Figures 3b and 3c/X.210 present alternative methods of indicating negative acknowledgements generated by the responding service-user. In Figure 3b/X.210 the same name (e.g., X) is used throughout the complete sequence, whereas in Figure 3c/X.210 the responding service-user employs a request with a different name (e.g., Y).



Note – Two styles of time-sequence diagrams are shown in Figure 3/X.210. Style 1 represents the passage of time by the angles of lines in the area representing the service provider. Style 2 represents the passage of time by the angles of arrows in areas representing service users. Each OSI Layer Service definition may adopt either of these styles for use in time-sequence diagrams. No difference in the sequence of primitives being described is implied by the style used.

FIGURE 3/X.210

Annotated time sequence diagrams

APPENDIX I

(to Recommendation X.210)

Conventions for naming service primitives

(This Appendix is not an integral part of the body of the Recommendation. It provides information for the authors of service Recommendations but is not necessary for users of service Recommendations.)

I.1 *Initials*

The following initials are used to specify Application services and the layers of the OSI Model:

a) P – Presentation Layer;

b) S – Session Layer;

c) T – Transport Layer;

d) N – Network Layer (see note 1);

e) DL – Data-Link Layer;

f) Ph – Physical Layer.

Note 1 – The use of 'N' to signify the Network Layer is not to be confused with the use of '(N)-' to signify a particular but unspecified layer of the Model.

Note 2 – Selection of initials for the various types of Application services requires further study.

I.2 *Service name*

A single word consisting of the infinitive form of a verb is recommended for the service name (e.g., CONNECT, ABORT).

I.3 *Name of primitive type*

The name of the primitive type consists of one of the following (indicating the type of the primitive):

a) request;

b) indication;

c) response (positive or negative);

d) confirm (positive or negative).

I.4 *Representation*

The initial(s) is represented in the form given in § I.1. The service name is written in capital letters and the name of the primitive type is written in lower case letters.

The initial(s) and the service name are separated by a hyphen; the service name and primitive type are separated by a space.

I.5 *Examples*

The following are examples of primitive names which use these conventions:

a) P-CONNECT request;

b) T-DATA indication;

c) S-DISCONNECT confirm.

APPENDIX II

(to Recommendation X.210)

Differences between Recommendation X.210 and ISO Technical Report 8509

II.1 CCITT Recommendation X.210 and ISO Technical Report 8509 are technically aligned to the extent that ISO layer service definitions using these conventions are not affected. However, there are a number of detailed wording differences that remain to be reconciled.

PHYSICAL SERVICE DEFINITION OF 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 (OSI) for CCITT Applications;

(b) that Recommendation X.210 defines the Service conventions for describing the Services of the layers of the OSI Reference Model;

Unanimously recommends

that this Recommendation provides the specification for the functions of the Physical Layer Service for operation in the Open Systems Interconnection environment using various transmission modes, topologies, and media.

CONTENTS

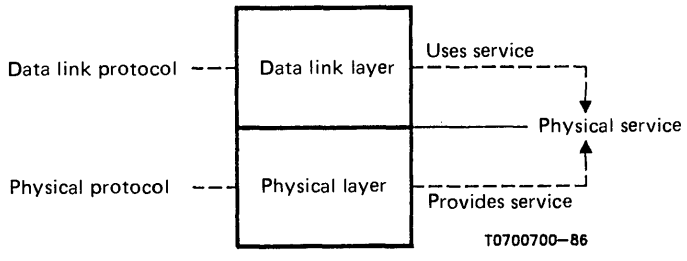
0 Introduction

0.1 *About this Recommendation*

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 Recommendation X.200. Reference Model of Open Systems Interconnection for CCITT Applications. The Reference Model of Open System Interconnection (OSI) subdivides the area of standardization for interconnection into a series of layers of specification, each of manageable size.

[1)] Recommendation X.211 and ISO/IEC 10022 [Information Processing Systems – Open Systems Interconnection – Physical Layer Service Definition] were developed in close collaboration and are technically aligned.

This Recommendation defines the Service provided by the Physical Layer to the Data Link Layer at the boundary between the Physical and Data Link layers of the OSI Reference Model. It provides for the designers of Data Link Protocols a definition of the Physical Service existing to support the Data Link Protocols and for the designers of Physical Protocols a definition of the Services to be made available through the action of the Physical Protocol over the underlying physical media, which are external to the OSI Physical Layer. The relationship of the Physical Layer with the Data Link Layer is illustrated in Figure 1/X.211.



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

FIGURE 1/X.211

Relationship of this Recommendation to other OSI Recommendations

1 Scope and field of application

This Recommendtion defines the OSI Physical Service in terms of:

a) the privimitive actions and events, of the Service;

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

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

The principal objective of this Recommendation is to specify the characteristics of a conceptual Physical Service and thus supplement the OSI Reference Model in guiding the development of Physical protocols.

This Recommendation does not specify individual implementations or products nor does it constrain the implementation of entities and interfaces within an information processing system.

There is no conformance of equipment to this Physical Service Definition Recommendation. Instead, conformance is achieved through implementation of conforming OSI Physical protocols that fulfil the Physical Service defined in this Recommendation.

2 References

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

2) Recommendation X.210 – Layer Service Definition Conventions of Open Systems Interconnection for CCITT Applications (see also ISO/TR 8509)

3 Definitions

3.1 *Reference Model Definitions*

This Recommendation is based on the concepts developed in the OSI Reference Model, Recommendation X.200, and makes use of the following terms defined in that Recommendation:

a) Data circuit
b) Physical connection
c) Physical Layer
d) Physical media
e) Physical Service
f) Physical Service access point
g) Physical Service data unit

3.2 *Service Conventions Definition*

This Recommendation also makes use of the following terms defined in Recommendation X.210, OSI Service Conventions, as they apply to the Physical Layer:

a) Physical Service user
b) Physical Service provider
c) primitive
d) request
e) indication

4 Abbreviations

OSI	Open Systems Interconnection
Ph	Physical
PhC	Physical connection
PhL	Physical Layer
PhS	Physical Service
PhSAP	Physical Service access point
PhSDU	Physical Service data unit
PhPDU	Physical Protocol data unit
QOS	Quality of Service

5 Conventions

5.1 *General Conventions*

This Recommendation uses the descriptive conventions given in Recommendation X.210. OSI Service Conventions.

The layer service model, service primitive, and time-sequence diagrams taken from those conventions are entirely abstract descriptions; they do not represent a specification for implementation.

5.2 *Parameters*

Service primitives, used to represent PhS-user/provider interactions (refer to Recommenation X.210) may convey parameters that indicate information available in the user/provider interaction.

The parameters, which apply to each group of Physical Service primitives, are set out in tables in Sections 11 to 14. The tables also indicate the association of which parameters can be carried by each respective primitive.

Some entries are further qualified by items in brackets. These may be:

a) an indication that the parameter is conditional in some way:

   (C) indicates that the parameter is not present on the primitive for every connection; the parameter definition describes the conditions under which the parameter is present or absent;

b) a parameter specific constraint:

(=) indicates that the value supplied in an indication primitive is always identical to that supplied in a previous request primitive issued at the peer service access point;

c) indication that some note applies to the entry:

(note x) indicates that the referenced note contains additional information pertaining to the parameter and its use.

In any particular interface, not all parameters used will be explicitly stated. Some may be implicitly associated with the PhSAP at which the primitive is issued.

5.3 *PhC Endpoint Identification Convention*

If at a PhSAP there is more than one PhC, and distinction among then is needed by the PhS user, PhC endpoint identification must be provided. All primitives issued at such a PhSAP would be required to use this mechanism to identify PhCs. Such an implicit identification is not described as a parameter of the service primitives in this Physical Service Definition.

When the PhC traverses relays, which are controlled through a separate PhC, an implicit identification mechanism must also provide for identification of these dependencies.

6 Overview and general characteristics

The Physical Service provides for the transparent transfer of data between PhS users. It makes invisible to the PhS users the way in which supporting communications resources are utilised to achieve this transfer. Service classes are defined to categorize the distinctions that are visible to the PhS user.

The PhS provides for a PhC between PhS users. Since connections cannot be established through the protocol at the Physical Layer but rather are configured when the service is created, the PhC, which is a logical concept, nevertheless must relate directly to the real physical media paths provided to the Physical Layer. For this reason:

a) there is no distinction between connection-mode and connectionless-mode at the Physical Layer. The service is independent of whether the higher layers operate in connection-mode or connectionless-mode;

b) each PhC is identified within the Physical Layer;

c) a PhC can only relate to a particular PhSAP (i.e., a PhC implies a specific source PhSAP and a specific destination PhSAP or group of PhSAPs if a multi-endpoint connection).

The PhC may traverse Physical Layer relay, or intermediate systems when several physical media are used in tandem. Such relaying may be controlled through a management function exercised over a separate, but related PhC, or may be controlled from the Network Layer, as specified in Recommendation X.200, § 7.5.4.1 for interconnection of data circuits. The Physical Layer does not make any routing decisions. Intermediate systems may also be used for mapping different Physical Layer protocols associated with a PhC.

Quality of Service provided by the Physical Service is predefined, in accordance with the class of service, though it may optionally be varied through management control of the configuration.

Actual data transmission takes place over the physical media. The mechanical, electromagnetic, and other media dependent characteristics of the physical media are defined at the boundary (interface) between the Physical Layer and the physical media. Definitions of these characteristics are found in other CCITT Recommendations and International Standards.

7 Features of the Physical Service

7.1 The Physical Service offers the following features to a PhS user:

a) the means to activate a PhC with another PhS user for the purpose of exchanging PhSDUs. More than one PhC may exist between the same pair of PhS users. The PhC Activate Service is optional and need not apply for duplex or simplex transmission (i.e. continuous Data Transfer Phase);

b) the means of transferring PhSDUs on a PhC. A PhSDU consists of one bit or a string of bits. PhSDUs are transferred in PhPDUs transparently over a PhC without change to the content (within the Quality of Service) or constraint on their data values. PhSDUs are delivered in the same order in which they are submitted;

c) the means to identify, when necessary, individual PhCs at the PhSAPs. Note that the parameters to identify a particular PhC within the PhSAP are implicit (see § 5.3);

d) the means for unconditional, and therefore possibly, destructive deactivation of a PhC by either the PhS user or by the PhS provider. The PhC Deactivate Service is optional and need not apply for duplex or simplex transmission (i.e. continuous Data Transfer Phase).

7.2 Other aspects of the Physical Service include:

a) the transfer of PhSDUs may be either duplex (two-way simultaneous), half-duplex (two-way alternate) or simplex (one way); either point-to-point or multi-endpoint; and either synchronous or asynchronous as appropriate (see § 8);

b) the data signalling rate on the physical media may not correspond with the PhSDU throughput rate due to the inclusion of Physical Layer protocol control information, multiplexing function, encoding mechanisms, or other transmission control functions;

c) PhSDU synchronization is provided by the Physical Service. This includes bit synchronization. Other delimiting may be available, which is a veritable feature;

d) PhC endpoint identifiers are not globally known. In the case of multiplexing, they will be conveyed implicitly via the Physical Layer protocol;

e) fault condition notification to the PhS user, beyond conveying a PhC deactivation indication, is for further study.

8 Classes of Physical Services

Distinctions of Physical Services are necessary to identify features that relate to the requirements of the service as seen by the Data Link Layer. These distinctions are:

a) Type of transmission – synchronous and asynchronous.

b) Mode of operation – duplex, half-duplex, and simplex.

Note – While these modes describe the operation at the Physical Layer Service boundary between the Physical Layer and the Data Link Layer, they do not necessarily imply the specific mode of operation of the Physical Layer Entity and the interface between the Physical Layer and the underlying physical media. This applies to operations associated with specific Physical Service Provider implementations, such as collision detection and multiplexing, which may map onto certain service primitives (for example, Activation and Deactivation) but are otherwise transparent to the Physical Service User.

9 Model of the Physical Service

9.1 *Model of the Layer Service*

This Recommendation uses the abstract model for a layer service defined in § 4 of Recommendation X.210, OSI Service Conventions. The model defines the interactions between the PhS users and the PhS provider that take place at PhSAPs. Information is passed between the PhS user and the PhS provider by service primitives, which may convey parameters. The description of the model is applicable to point-to-point PhCs (linking two PhSAPs). The extension of the model for multi-endpoint PhCs is for further study.

9.2 *Model of a physical connection*

The operation of a PhC is modeled in the abstract by a pair of bit streams linking the two PhSAPs. There is one bit stream for each direction of transmission (see Figure 2/X.211). Each bit stream conveys Physical Protocol Data Units (PhPDUs). Bits within each bit stream are delivered in the same order in which they were submitted.

9.3 *Model of a relayed point-to-point PhC where the relay is controlled within the PhS Provider*

The operation of a PhC is modeled exactly as described in § 9.2 except for the interposition of the relay within the data circuit to support tandem physical media (see Figure 3/X.211).

9.4 *Model of a relayed point-to-point PhC where the relay is controlled from the Network Layer*

The operation of each of the relay-controlling PhCs can be accomplished by the Network Layer protocol control information being conveyed either via the same PhC (in-band signalling), or via a separate PhC (out-of-band signalling), see Figure 4/X.211. Physical Layer relay systems do not complete the end-to-end PhC until Network Layer control actions are completed among the Network Layer entities en route. Deactivation may be accomplished through either Network layer protocol or management actions.

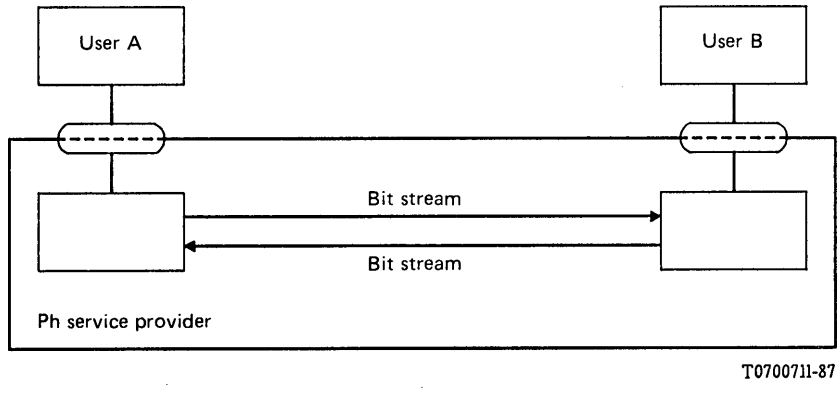


FIGURE 2/X.211

Simple model of a PhC

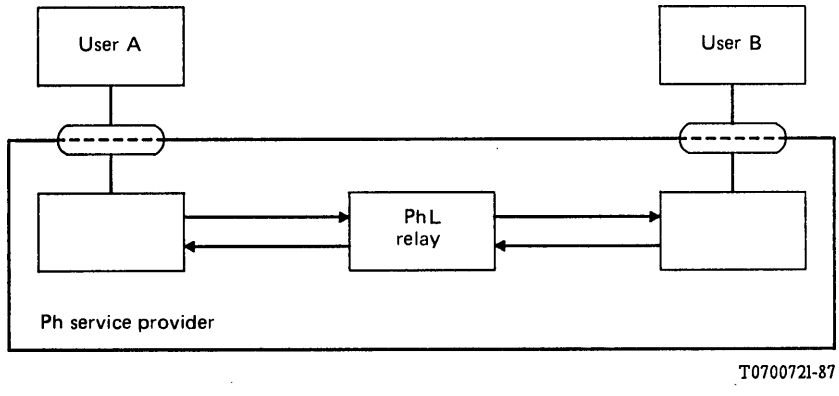
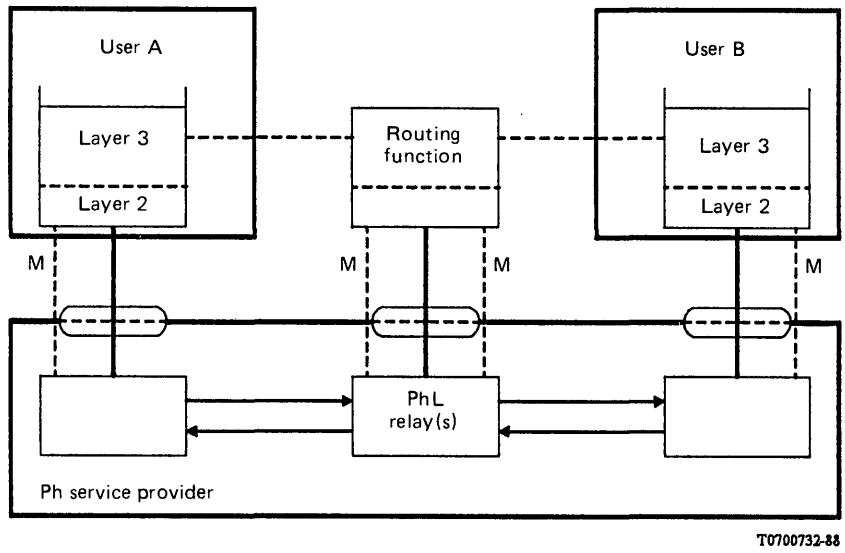


FIGURE 3/X.211

Simple model of a PhC relayed within PhS Provider



M = Management functions

FIGURE 4/X.211

Simplified model of a PhC relayed with control from the network layer

10 Quality of Physical Service

The term "Quality of Service" (QOS) refers to certain characteristics of a PhC as observed between the PhC endpoints. QOS describes aspects of a PhC that are attributable solely to the PhS provider: it can only be properly determined in the absence of PhS user behaviour (which is beyond the control of the PhS provider) that specifically constrains or impairs the performance of the Physical Service.

The PhS users have knowledge of the relevant QOS of the PhC. This is true even in the case of a PhC scanning several physical circuits.

The Quality of Service of a PhC is dependent on the physical media of interconnection. It may be characterized by:

a) service availability;

b) error rate, where errors may arise from alteration, loss, creation, and other causes;

c) throughput;

d) transit delay;

e) protection (e.g. encryption).

QOS is described in terms of QOS parameters. These parameters are selected and determined by methods other than Physical Service primitives, although they may be determined in some cases through layer management primitives.

There is no guarantee that the originally agreed upon QOS values will be maintained throughout the lifetime of the PhC. The PhS users should be aware that a change in QOS is not explicitly signalled in the Physical Service, although in some cases it may be signalled through layer management primitives.

11 Sequence of primitives

This section defines the constraints on the sequences in which the primitives defined in §§ 12-14 may occur. The constraints determine the order in which primitives occur, but do not fully specify when they may occur. Table 1/X.211 is a summary of the PhS primitives and their parameters, and defines the phases in which they occur (Activation, Data Transfer, and Deactivation).

TABLE 1/X.211

Summary of physical service primitives and parameters

Phase	Service	Primitive	Parameters
PhC activation (Note 1)	PhC activation	Ph-ACTIVATE request ..... Ph-ACTIVATE indication	(Note 2)
Data transfer	Data transfer	Ph-DATA request ..... Ph-DATA indication	PhS-User data
PhC deactivation (Note 1)	PhC deactivation	Ph-DEACTIVATE request ..... Ph-DEACTIVATE indication	(Note 3)

Note 1 – The PhC activation and the PhC deactivation services are optional and need not apply for duplex or simplex transmission.

Note 2 – Parameters associated with the classes of service (see Section 8) are for further study.

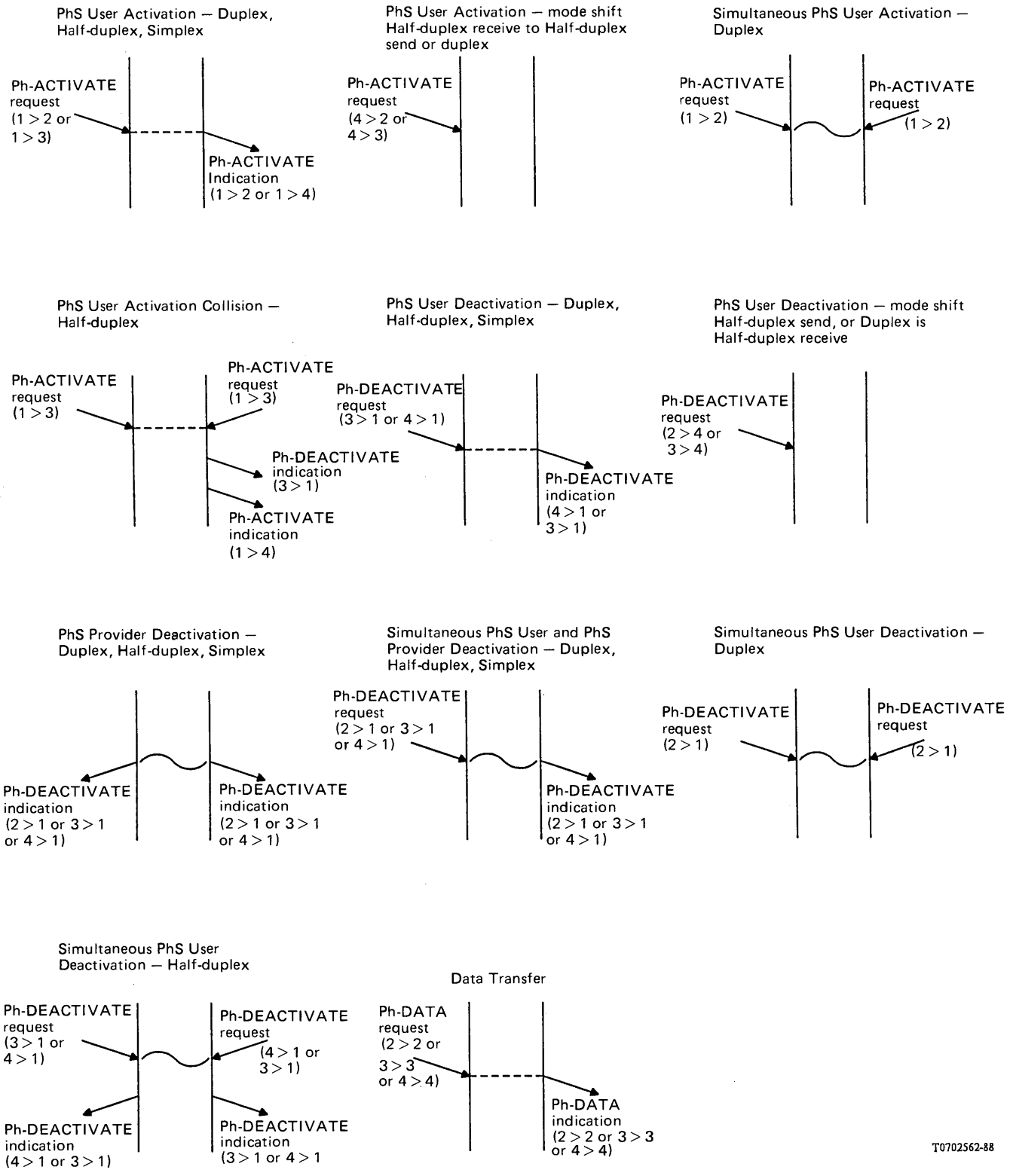
Note 3 – Parameters associated with PhC deactivation are for further study.

11.1 *Relation of primitives at the two PhC endpoints*

A primitive issued at one PhC endpoint will, in general, have consequences at the other PhC endpoint. The relations of primitives of each type to primitives at the other PhC endpoint are defined in the appropriate subsection in §§ 12-14; these relations are summarized in the diagrams in Figure 5/X.211. Additional sequences and relationships are for further study.

11.2 *Sequence of primitives at one PhC endpoint*

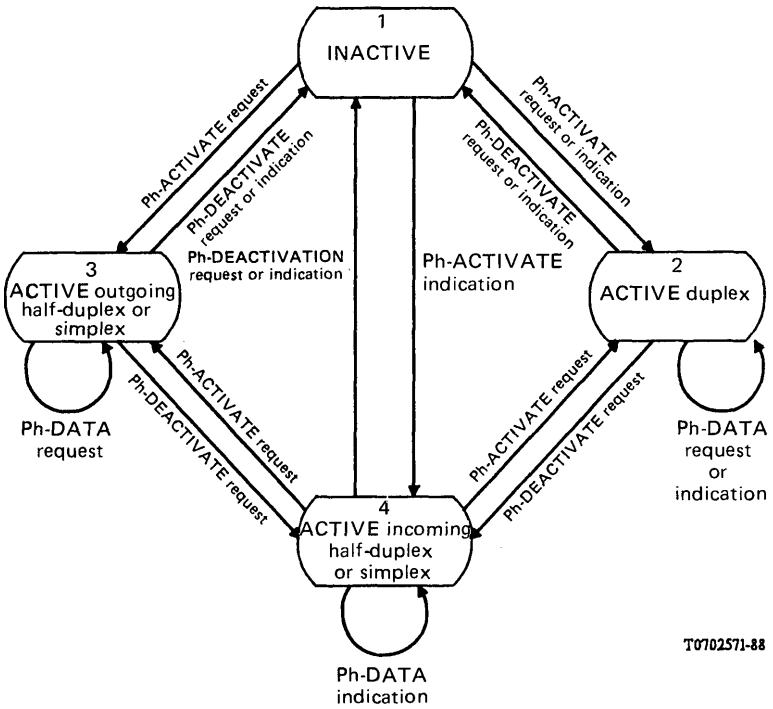
The recognized sequence of primitives at PhC endpoints is defined in the composite state transition diagram, Figure 6/X.211. Specific primitive sequences that apply to individual modes of operations and topologies are shown in Figures 6a/X.211 through 6i)/X.211.



Note – Numbers in brackets refer to the applicable transitions between the states indicated in Figure 6/X.211 for the related PhC Endpoint.

FIGURE 5/X.211

Summary of physical service primitive time sequence to diagrams



Note – The parameters associated with the Activation and Deactivation primitives (which are for further study) will differentiate the constituent elements of this composite diagram such that the particular state transition sequences in Figures 6a to 6i (only one of which need be provided) are distinguished. The detailed relationship between these distinctions and those in clause 8 is also for further study.

FIGURE 6/X.211

Composite state transition diagram for sequences of primitives at a PhC Endpoint

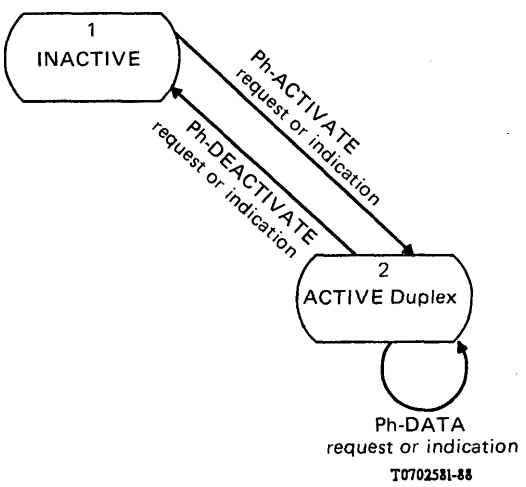


FIGURE 6a/X.211

State transition diagram for duplex mode with Activation/Deactivation

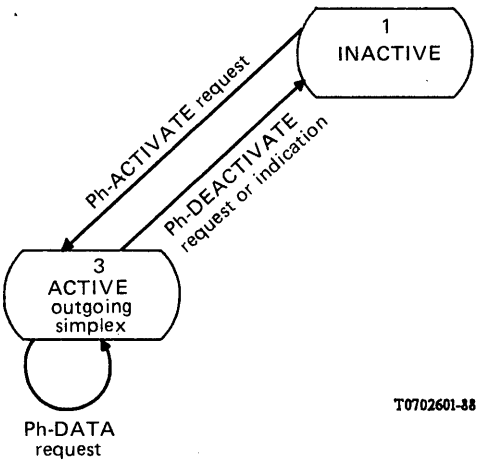


FIGURE 6b/X.211

State transition diagram for simplex mode, outgoing

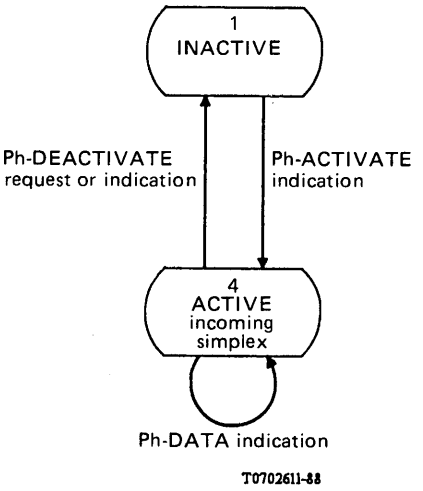


FIGURE 6c/X.211

State transition diagram for simplex mode, incoming

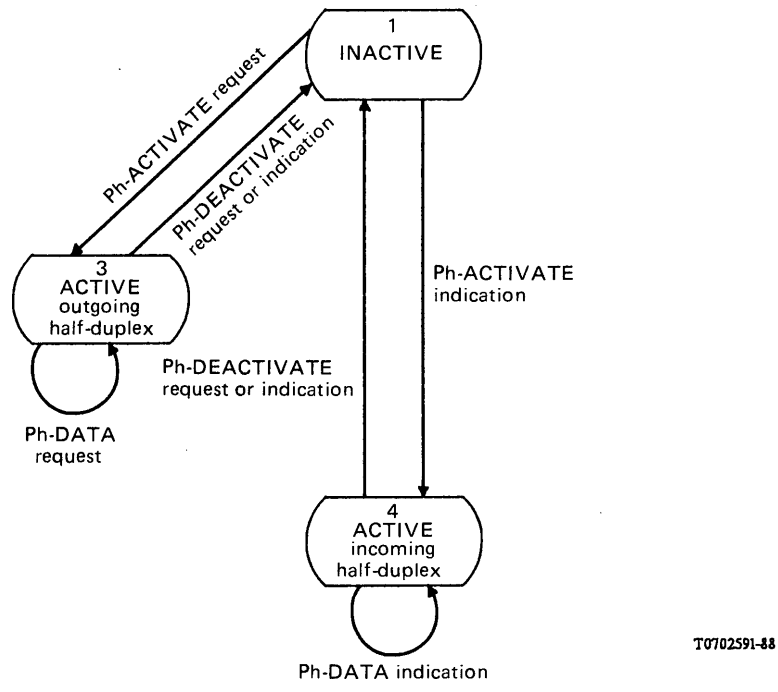


FIGURE 6d/X.211

State transition diagram for half-duplex mode

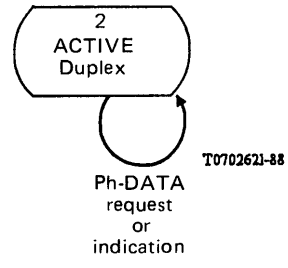


FIGURE 6e/X.211

State transition diagram for duplex mode, data transfer phase only

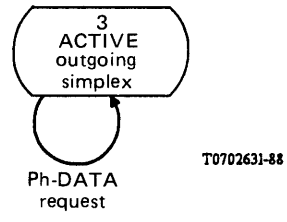


FIGURE 6f/X.211

State transition diagram for simplex mode, outgoing, data transfer phase only

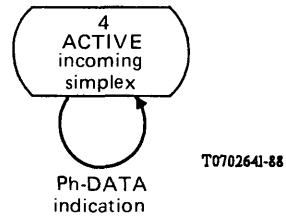


FIGURE 6g/X.211

State transition diagram for simplex mode, incoming, data transfer phase only

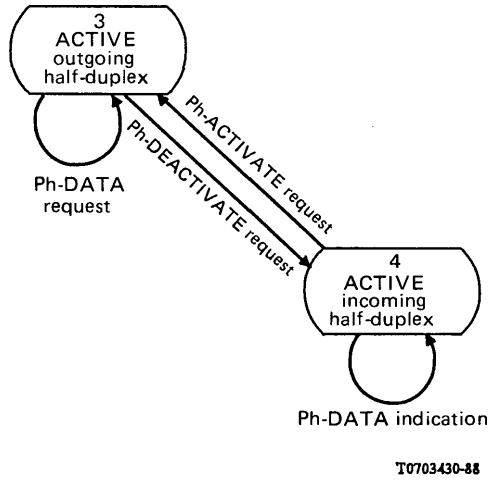


FIGURE 6h/X.211

State transition diagram for half-duplex/duplex without inactive state

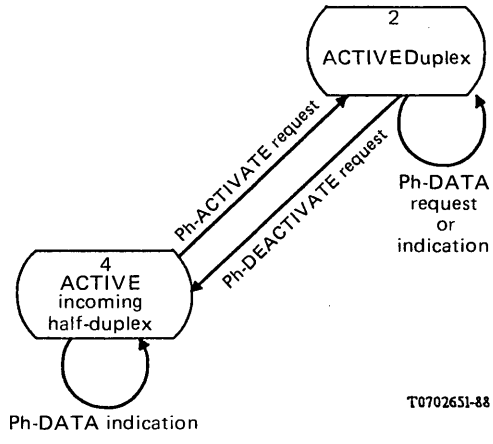


FIGURE 6i/X.211

State transition diagram for half-duplex/duplex mode shift

12 PhC activation phase

12.1 *Function*

The PhC activation service primitives are used for activating directions of PhPDU transmission. They are required for half-duplex and are optional for duplex and simplex. The Ph-ACTIVATE request primitive requests activation of the PhC. Each direction of transmission is activated independently for half-duplex operation, and both directions of transmission are activated for duplex operation. For half-duplex and simplex operation, the Ph-ACTIVATE request primitive activates the outgoing direction of transmission, and the Ph-ACTIVATE indication primitive indicates activation of the incoming direction of transmission. During half-duplex operation, a Ph-ACTIVATE request cannot be issued by the PhS user after receipt of the Ph-ACTIVATE indication and before the receipt of a Ph-DEACTIVATE indication primitive.

12.2 *Types of primitives and parameters*

The PhC activation service involves two primitives as shown in Table 2/X.211. The parameters in the table are for further study.

TABLE 2/X.211

Physical service activate primitives and parameters

Primitive / Parameter	Ph-ACTIVATE request	Ph-ACTIVATE indication
(Note)		

Note – Parameters associated with the classes of service (see Section 8) are for further study.

12.3 *Sequence of primitives*

The sequence of primitives in a successful activation of a direction of transmission is defined by the time sequence diagram in Figure 7/X.211.

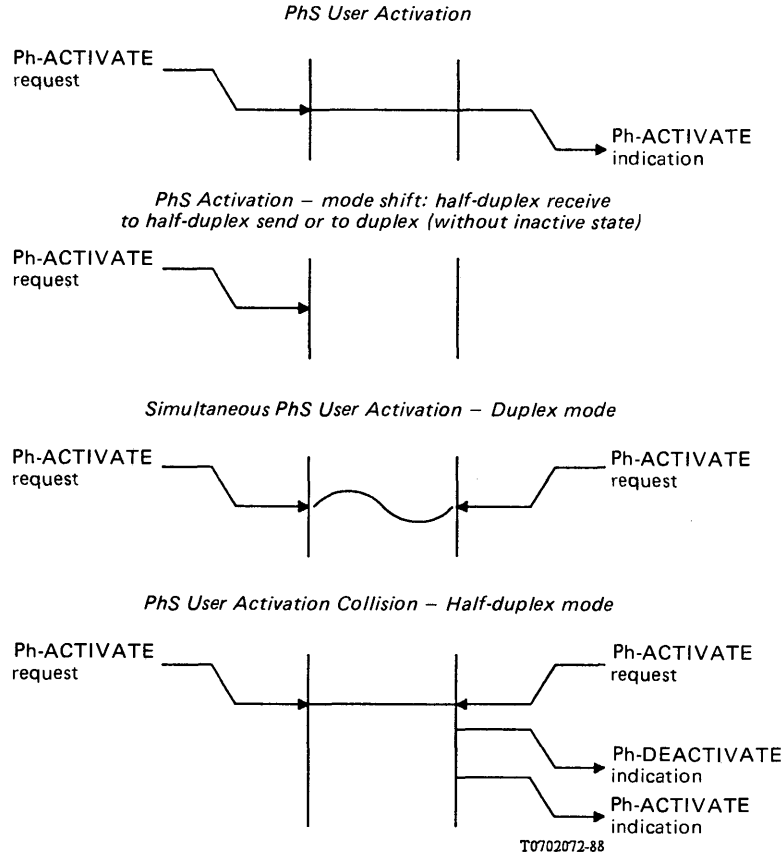
13 PhC deactivation phase

13.1 *Function*

The PhC deactivation service primitives are used for deactivating direction of PhPDU transmission. They are required for half-duplex and are optional for duplex and simplex. The Ph-DEACTIVATE request primitive requests deactivation of the PhC. Each direction of transmission is deactivated independently for half-duplex operation, and both directions of transmission are deactivated for duplex operation. For half-duplex and simplex operation, the Ph-DEACTIVATE request primitive deactivates the outgoing direction of transmission, and the Ph-DEACTIVATE indication primitive indicates deactivation of the incoming direction of transmission. During half-duplex operation, a Ph-ACTIVATE request primitive can be issued by a PhS user after receipt of the Ph-DEACTIVATE indication primitive.

13.2 *Types of primitives and parameters*

The PhC deactivation service involves two primitives as shown in Table 3/X.211.



Note – There is no indication to the PhS user, who issued the ACTIVATE request, when the PhS provider is unable to activate the direction of transmission.

FIGURE 7/X.211

Sequence of primitives for activation

TABLE 3/X.211

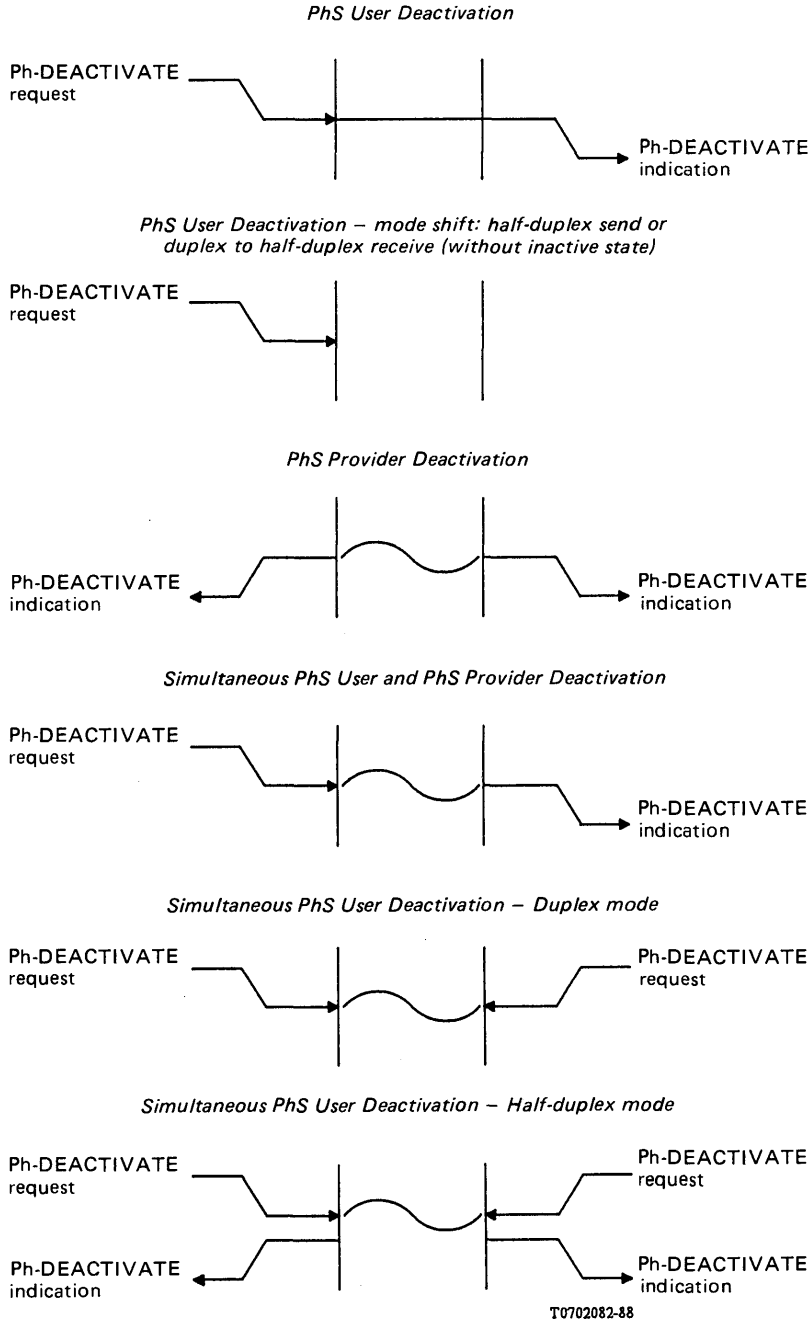
Physical service deactivate primitives and parameters

Parameter \ Primitive	Ph-DEACTIVATE request	Ph-DEACTIVATE indication
(Note)		

Note – The need for deactivation parameters, e.g., Originator, are for further study. The Originator parameter indicates the source of the PhC deactivation. Its value indicates whether PhS user, PhS provider, or unknown caused the action.

13.3 *Sequence of primitives*

The sequence of primitives for PhC deactivation are expressed in the time sequence diagrams in Figure 8/X.211.



Note – The Ph-DEACTIVATE indication might not distinguish between invocation by the remote PhS user or invocation by the PhS provider (e.g., due to a failure).

FIGURE 8/X.211

Sequence of primitives for deactivation

14 Data transfer phase

14.1 *Function*

The data transfer service primitives provide for an exchange of user data called Physical Service Data Units (PhSDUs). The Physical Service preserves the sequence of the PhSDUs.

14.2 *Types of primitives and parameters*

Table 4/X.211 indicates the types of primitives and the parameters needed for data transfer.

TABLE 4/X.211

Data transfer primitives and parameters

Parameter \ Primitive	Ph-DATA request	Ph-DATA indication
User data	X	X(=)

The User Data parameter conveys PhSDUs for transmission between the PhS users. The size of the PhSDU is a PhS provider option. The PhS user has a prior knowledge of the PhSDU size value.

14.3 *Sequence of primitives*

The operation of the Physical Service in transferring PhSDUs can be modeled as a pair of bit streams within the PhS provider (see § 9).

The Physical Layer serves to pass a transparent bit stream (PhSDUs) continuously. This requires that the Ph-DATA primitives are present, as necessary, throughout the Data Transfer Phase. Immediately upon receipt of a Ph-ACTIVATE indication primitive, an incoming user data bit stream (PhSDUs) is available to the PhS user. Upon issuance of a Ph-ACTIVATE request primitive, an outgoing user data bit stream (PhSDUs) is assumed to be available to the PhS user.

The sequence of primitives in a successful data transfer is defined in the time sequence diagram in Figure 9/X.211.

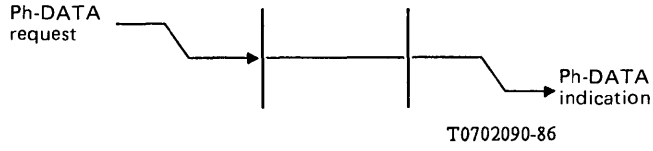


FIGURE 9/X.211

Sequence of primitives in data transfer

The sequence of primitives in Figure 9/X.211 may remain uncompleted if an Ph-DEACTIVATE primitive occurs.

DATA LINK SERVICE DEFINITION 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.210 specifies the OSI Layer Service Definition Conventions for describing the services of layers of the OSI Reference Model,

unanimously recommends

(1) that the scope and field of application of the Open Systems Interconnection Data Link Service Definition are given in § 1;

(2) that the general requirements of the Open Systems Interconnection Data Link Service are given in Part 1;

(3) that the connection-mode Data Link Service is defined in Part 2;

(4) that the connectionless-mode Data Link Service is defined in Part 3.

CONTENTS

[1)] Recommendation X.212 and ISO 8886 [Information Procesing Systems – Data Communications – Data Link Service Defintion] were developed in close collaboration and are technically aligned, except for the differences noted in Appendix I.

0 Introduction

0.1 *About this Recommendation*

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 Recommendation X.200, Reference Model of Open Systems Interconnection for CCITT Applications. The reference model described by Recommendation X.200 subdivides the area of standardization for Open Systems Interconnection (OSI) into a series of layers of specification, each of a manageable size.

This Recommendation defines the services provided by the Data Link Layer to the Network Layer at the boundary between the Data Link and Network Layers of the OSI Reference Model. It provides for the designers of network protocols a definition of the Data Link Service existing to support the network protocol and for the designers of Data Link Protocols a definition of the services to be made available through the action of the Data Link Protocol over ther underlying service. The relationship is illustrated in Figure 1/X.212.

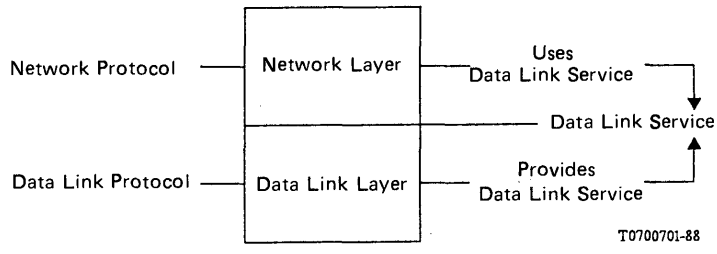


FIGURE 1/X.212

Relationship of this Recommendation to other OSI Recommendations

Throughout the set of OSI Recommendations, the term "service" refers to the abstract capability provided by one layer of the OSI Reference Model to the layer immediately above. Thus, the Data Link Service defined in this document is a conceptual architectural service, independent of administrative divisions.

1 Scope and field of application

This Recommendation defines the OSI Data Link Service 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 that they take; and

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

The principal objective of this Recommendation is to specify the characteristics of a conceptual Data Link Service and thus, supplement the OSI Reference Model in guiding the development of Data Link Protocols.

This Recommendation does not specify individual implementation or products, nor does it constrain the implementation of Data Link entities and interfaces within an information processing system.

There is no conformance of equipment to this Data Link Service Definition Recommendation. Instead, conformance is achieved through implementation of conforming Data Link Protocols that fulfil the Data Link Service defined in this Recommendation.

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 TR8509).

PART ONE – GENERAL

3 Definitions

3.1 *OSI Reference Model Definitions*

This Recommendation is based on the concepts developed and makes use of the following terms defined in Recommendation X.200:

a) Data link entity
b) Data Link Layer
c) Data-link-Service
d) Data-link-service-access-point
e) Data-link-service-access-point-address
f) Data-link-service-data-unit
g) Reset.

3.2 *Service Conventions Definitions*

This Recommendation makes use of the following terms defined in Recommendation X.210, as they apply to the Data Link Layer:

a) Data Link Service User
b) Data Link Service Provider
c) Primitive
d) Request
e) Indication
f) Response
g) Confirm

3.3 *Data Link Service Definitions*

This Recommendation makes use of the following terms:

a) **data-link-connection**

An association established by a Data Link Layer between two or more Data Link Service users for the transfer of data, which provides explicit identification of a set of Data Link data transmissions and agreement concerning the Data Link data transmission services to be provided for the set.

(*Note* – This definition clarifies the definition given in X.200.)

b) **data-link-connection-mode data transmission**

The transmission of a Data-link-service-data-unit within the context of a Data-link-connection that has been previously established.

c) **data-link-connectionless-mode data transmission**

The transmission of a Data-link-service-data-unit not in the context of a Data-link-connection and not required to maintain any logical relationship among multiple invocations.

4 Abbreviations

DL	Data Link
DLC	Data-link-connection
DLL	Data Link Layer
DLS	Data Link Service
DLSAP	Data-link-service-access-point
DLSDU	Data-link-service-data-unit
OSI	Open Systems Interconnection
QOS	Quality of Service

5 Conventions

5.1 *General conventions*

This Recommendation uses the descriptive conventions given in Recommendation X.210.

The service model, service primitives, and time-sequence diagrams used are entirely abstract descriptions; they do not represent a specification for implementation.

5.2 *Parameters*

Service primitives, used to represent service user/service provider interactions (Recommendation X.210), convey parameters which indicate information available in the user/provider interaction.

The parameters which apply to each group of Data Link Service primitives are set out in tables in sections 12 to 14 and 19. Each "X" in the tables indicates that the primitive labelling the column in which it falls may carry the parameter labelling the row in which it falls.

Some entries are further qualified by items in brackets. These may be:

a) A parameter specific constraint:

(*) indicates that the value supplied in an indication or confirm primitive is always identical to that supplied in a previous request or response primitive issued at the peer service-access-point

b) Indication that some note applies to the entry:

(see note X) indicates that the referenced Note contains additional information pertaining to the parameter and its use.

In any particular interface, not all parameters need be explicitly stated. Some may be implicitly associated with the DLSAP at which the primitive is issued.

6 Overview of the Data Link Service

The DLS provides for the transparent and reliable transfer of data between DLS users. It makes invisible to these DLS users the way in which supporting communications resources are utilized to achieve this transfer.

In particular, the DLS provides for the following:

a) Independence of underlying Physical Layer – the DLS relieves DLS users from all concerns regarding which configuration is available (e.g. point-to-point connection) or which physical facilities are used (e.g. half-duplex transmission).

b) Transparency of transferred information – the DLS provides for the transparent transfer of DLS user-data. It does not restrict the content, format or coding of the information, nor does it ever need to interpret its structure or meaning.

c) Reliable transfer of data – the DLS relieves the DLS user from loss, insertion, corruption or, if requested, misordering of data which may occur. In some cases of unrecoverable errors in the Data Link Layer, duplication or loss of DLSDUs may occur.

Note – Detection of duplicate or lost DLSDUs may be performed by DLS users.

d) Quality of Service selection – the DLS makes available to DLS users a means to request and to agree upon a quality of service for the transfer of data. QOS is specified by means of QOS parameters representing characteristics such as throughput, transit delay, accuracy and reliability.

e) Addressing: The DLS allows the DLS user to identify itself and to specify the DLSAP to which a DLS is to be established whenever more than two DLSAPs are supported by the DLS provider. Data Link addresses have only local significance within a specific Data Link configuration over a single transmission medium (point-to-point or multi-point physical connection) or a group of parallel transmission media (multi-link or splitting function). Therefore it is not appopriate to define a global addressing structure.

Note – The DLS is required to differentiate between the individual systems that are physically or logically connected to a multipoint data link and to differentiate between connections when the data link layer includes a multiplexing function. For commonality with other Service definitions, this mechanism is referred to as addressing and the objects used to differentiate between systems are referred to as addresses.

7 Classes and types of Data Link Service

There are no distinct classes of Data Link Service defined.

There are two types of Data Link Service:

a) a connection-mode service (defined in Part 2); and

b) a connectionless-mode service (defined in Part 3).

When making reference to this Recommendation, a user or provider of Data Link Service shall state which types of service it expects to use or provide.

PART TWO – DEFINITION OF THE CONNECTION-MODE SERVICE

8 Features of the connection-mode Data Link Service

The DLS provides the following features to the DLS user:

a) the means to establish a DLC with another DLS user for the purpose of exchanging DLSDUs;

b) the establishment of an agreement between the initiating DLS user and the DLS provider for a certain QOS associated with each DLC;

c) the means of transferring DLSDUs of restricted length on a DLC. The transfer of DLSDUs is transparent, in that the boundaries of DLSDUs and the contents of DLSDUs are preserved unchanged by the DLS, and there are no constraints on the DLSDU content imposed by the DLS;

Note – The length of a DLSDU may be limited because of internal mechanisms employed by the data-link-protocol (see Recommendation X.200 § 7.6.3.2).

d) the means by which the receiving DLS user may flow control the rate at which the sending DLS user may send DLSDUs;

e) the means by which a DLC can be returned to a defined state and the activities of the two DLS users synchronized by use of a Reset service element;

f) the unconditional, and therefore possibly destructive, release of a DLC by either of the DLS users or by the DLS provider.

9 Model of the connection-mode Data Link Service

This Recommendation uses the abstract model for a layer service defined in section 4 of Recommendation X.210. The model defines the interactions between the DLS users and the DLS provider which take place at the two DLSAPs. Information is passed between the DLS user and the DLS provider by service primitives, which may convey parameters.

9.1 *DLC Endpoint Connection Identification*

If a DLS user needs to distinguish among several DLCs at the same DLSAP, then a local connection endpoint identification mechanism must be provided. All primitives issued at such a DLSAP within the context of a DLC would be required to use this mechanism to identify this DLC. Such an implicit identification is not described in this Recommendation.

9.2 *Model of a Data-link-connection*

Between the two endpoints of a DLC, there exists a flow control function that relates the behaviour of the DLS user receiving data to the ability of the other DLS user to send data. As a means of specifying this flow control feature and its relationship with other capabilities provided by the connection-mode DLS the queue model of a DLC, which is described in the following sections, is used.

This queue model of a DLC is discussed only to aid in the understanding of the end-to-end service features perceived by DLS users. It is not intended to serve as a substitute for a precise, formal description of the DLS, nor as a complete specification of all allowable sequences of DLS primitives. (Allowable primitive sequences are specified in section 11. See also the note below.) In addition, this model does not attempt to describe all the functions or operations of DL entities that are used to provide the DLS. No attempt to specify or constrain DLS implementations is implied.

Note – The internal mechanisms which support the operation of the DLS are not visible to the DLS user. In addition to the interactions between service primitives described by this model (e.g. the issue of a DL-Reset request at an DLSAP may prevent the receipt of a DL-DATA indication, corresponding to a previously issued DL-DATA request, by the peer DLS user) there may also be:

a) constraints applied locally on the ability to invoke primitives;

b) service procedures defining particular sequencing constraints on some primitives.

9.2.1 *Queue model concepts*

The queue model represents the operation of a DLC in the abstract by a pair of queues linking the two DLSAPs. There is one queue for each direction of information flow (see Figure 2/X.212).

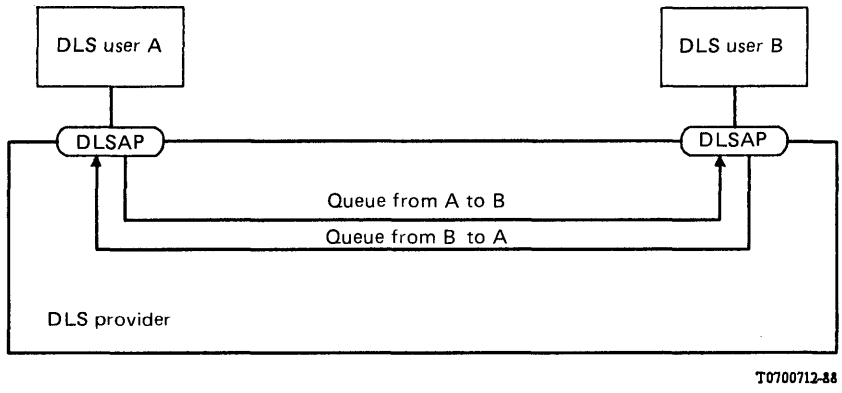


FIGURE 2/X.212

Queue model of a DLC

Each queue represents a flow control function in one direction of transfer. The ability of a DLS user to add objects to a queue will be determined by the behaviour of the other DLS queue. Objects are entered or removed from the queue as a result of interactions at the two DLSAPs.

The pair of queues is considered to be available for each potential DLC.

The following objects may be placed in a queue by a DLS user (see sections 12-14):

a) a connect object, representing a DL-CONNECT primitive and its parameters;

b) a data object, representing a DL-DATA primitive and its parameters;

c) a reset object, representing a DL-RESET primitive and its parameters; and

d) a disconnect object, representing a DL-DISCONNECT primitive and its parameters.

The following objects may be placed in a queue by the DLS-provider (see sections 12-14):

1) a reset object, representing a DL-RESET primitive and its parameters;

2) a synchronization mark object (see § 9.2.4); and

3) a disconnect object, representing a DL-DISCONNECT primitive and its parameters.

The queues are defined to have the following general properties:

i) a queue is empty before a connect object has been entered and can be returned to this state, with loss of its contents, by the DLS provider;

ii) objects are entered into a queue by the sending DLS user, subject to control by the DLS provider. Objects may also be entered by the DLS provider;

iii) objects are removed from the queue, under the control of the receiving DLS user;

iv) objects are normally removed in the same order that they were entered (however, see § 9.2.3); and

v) a queue has a limited capacity, but this capacity is not necessarily either fixed or determinable.

9.2.2 *DLC Establishment*

A pair of queues is associated with a DLC between two DLSAPs when the DLS provider receives a DL-CONNECT request primitive at one of the DLSAPs, and a connect object is entered into one of the queues. From the standpoint of the DLS users of the DLC, the queues remain associated with the DLC until a disconnect object representing a DL-DISCONNECT primitive is either entered or removed from the queue.

DLS user A, who initiates a DLC establishment by entering a connect object representing a DL-CONNECT request primitive into the queue from DLS user A to DLS user B, is not allowed to enter any other object, other than a disconnect object, into the queue until after the connect object representing the DL-CONNECT confirm primitive has been removed from the DLS user B to DLA user A queue. In the queue from DLS user B to DLS user A objects can be entered only after DLS user B has entered a connect object representing a DL-CONNECT response primitive.

The properties exhibited by the queues while the DLC exists represent the agreements reached among the DLS users and the DLS provider during this connection establishment procedure concerning QOS.

9.2.3 *Data transfer*

Flow control on the DLC is respresented in this queue model by the management of the queue capacity, allowing objects to be added to the queues. The addition of an object may prevent addition of a further object.

Once objects are in the queue, the DLS provider may manipulate pairs of adjacent objects, resulting in deletion. An object may be deleted if, and only if, the object which follows it is defined to be destructive with respect to the object. If necessary the last object in the queue will be deleted to allow a destructive object to be entered – they may therefore always be added to the queue. Disconnect objects are defined to be destructive with respect to all other objects. Reset objects are defined to be destructive with respect to all other objects except connect, disconnect objects.

The relationships between objects which may be manipulated in the above fashion are summarized in Table 1/X.212.

Whether the DLS provider performs actions resulting in deletion or not will depend upon the bahaviour of the DLC users and the agreed QOS for the DLC. In general, if a DLS user does not remove objects from a queue, the DLS provider shall, after some unspecified period of time, perform all the permitted deletions.

TABLE 1/X.212

Relationships between queue model objects

The following object y is defined / with respect to the preceding object x	Connect	Data	Reset	Synchronization mark	Disconnect
Connect	N/A	–	–	N/A	DES
Data	N/A	–	DES	N/A	DES
Reset	N/A	–	DES	–	DES
Synchronization mark	N/A	–	DES	N/A	DES
Disconnect	N/A	N/A	N/A	N/A	DES

N/A: x will not precede y in a valid state of a queue

– : Not to be destructive nor to be able to advance ahead

DES: To be destructive to the preceding object

9.2.4 *Reset*

In order to accurately model the reset service a synchronization mark object is required. The synchronization mark object exhibits the following properties:

a) it cannot be removed from a queue by a DLS-user;

b) a queue appears empty to a DLS-user when a synchronization mark object is the next object in the queue;

c) a synchronization mark object can be destroyed by a Disconnect object (see Table 1/X.212);

d) when a Reset object is immediately preceded by a synchronization mark object, both the Reset object and the synchronization mark object are deleted from the queue.

The initiation of a reset procedure is represented in the two queues as follows:

i) initiation of a reset procedure by the DLS provider is represented by the introduction into each queue of a reset object followed by a synchronization mark object;

ii) a reset procedure initiated by a DLS user is represented by the addition, by the DLS provider, of a reset object into the queue from the Reset initiator to the peer DLS user and the insertion of a reset object followed by a synchronization mark object into the other queue.

Unless destroyed by a disconnect object, a synchronization mark object remains in the queue until the next objet following it in the queue is a reset object. Both the synchronization mark object and the following reset object are then deleted by the DLS provider.

Note – Associated with the initiation of a reset procedure are restrictions on the issuance of certain other types of primitives. These restrictions will result in restrictions on the entry of certain object types into the queue until the reset procedure is completed (see § 14.2.3).

9.2.5 *DLC release*

The insertion into a queue of a disconnect object, which may occur at any time, represents the initiation of a DLC release procedure. The release procedure may be destructive with respect to other objects in the two queue and eventually results in the emptying of the queues and the disassociation of the queues with the DLC.

The insertion of a disconnect object may also represent the rejection of a DLC establishment attempt or the failure to complete DLC establishment. In such cases, if a connect object representing a DL-CONNECT request primitive is deleted by a disconnect object, then the disconnect object is also deleted. The disconnect object is not deleted when it deletes any other object, including the case where it deletes a connect object representing a DL-CONNECT response.

10 Quality of connection-mode Data Link Service

The term "Quality of Service" (QOS) refers to certain characteristics of a DLC as observed between the connection endpoints. QOS describes aspects of a DLC that are attributable solely to the DLS provider.

Once a DLC is established, the DLS users at the two ends have the same knowledge and understanding of what the QOS over the DLC is.

10.1 *Determination of QOS for Connection-mode Service*

QOS is determined in terms of QOS parameters. These parameters give DLS users a method of specifying their needs and give the DLS provider a basis for protocol selection.

The DLS QOS parameters can be divided into the following two types, based upon the way in which their values are determined:

a) those QOS parameters which may be selected on a per-connection basis during the establishment phase of a DLC;

b) those QOS parameters which are not selected during DLC establishment but whose values are known by other methods.

There are three QOS parameters, throughput, protection, and priority (as defined in §§ 10.2.1, 10.2.5 and 10.2.6 respectively), which are of the type that may be selected during DLC establishment. The selection procedures for these parameters are described in detail in § 12.2.5. Once the DLC is established, throughout the lifetime of the DLC, the agreed QOS values are not reselected at any point, and there is no guarantee that the original values will be maintained. The DLS users should also be aware that changes in QOS on a DLC are not explicity signalled by the DLS provider.

The remaining QOS characteristics that are identified as parameters, but for which there is no selection during DLC establishment, are defined in §§ 10.2.2 through 10.2.4 respectively. The values of these parameters for a particular DLC are determined by other methods such as *a priori* knowledge and agreement.

If selection is allowed, certain measures of QOS are requested by the sending DLS user when the DL-CON primitive action is initiated. The requested measures (or parameter values and options) are based on *a priori* knowledge by the DLS user of the service(s) made available to it by the DLS provider. Knowledge of the characteristics and type of service provided (i.e. the parameters, formats, and options that affect the transfer of data) is made available to a DLS user through some layer management interaction prior to (any) invocation of the DL connection-mode service. Thus, the DLS user has explicit knowledge of the characteristics of the service it can expect to be provided with each invocation of the service.

The DLS provider may also provide information on the current QOS independently of access to the service by the DLS user. This seemingly dynamic aspect of QOS determination is not a negotiation but provided with knowledge of the characteristics of the service currently outside of any instance of the invocation of the service.

10.2 *Definition of connection-mode QOS parameters*

QOS parameters can be classified as:

a) Parameters that express DLS performance, as shown in Table 2/X.212.

TABLE 2/X.212

Classification of performance QOS parameters

Performance criterion	
Speed	Accuracy reliability
Throughput transit delay	Residual error rate (corruption, duplication/loss) Resilience

b) parameters that express other DLS characteristics, as shown in Table 3/X.212.

TABLE 3/X.212

QOS parameters not associated with performance

Protection
Priority

Note – Some QOS parameters are defined in terms of the issuance of DLS primitives. Reference to a DLS primitive refers to the complete execution of that DLS primitive at the appropriate DLSAP.

10.2.1 *Throughput*

Throughput is defined as the total number of DLSDU bits successfully transferred by a DL-DATA request/DL-DATA indication primitive sequence divided by the input/output time for that sequence.

Successful transfer of the bits in a transmitted DLSDU is defined to occur when the bits are delivered to the intended receiving DLS user without error, in the proper sequence, prior to release of the DLC by the receiving DLS user.

The input/output time for a DL-DATA request/DL-DATA indication primitive sequence is the greater of the two times in the following list:

a) the time between the first and the last DL-DATA request in the sequence;

b) the time between the first and the last DL-DATA indication in the sequence.

Throughput is only meaningful for a sequence of complete DLSDUs.

Throughput is specified independently for each direction of transfer. In general, each throughput specification will define both the desired target value and the minimum acceptable value (or lowest acceptable QOS) for a DLC. Each specification is an average rate and will be based on a previously stated average DLSDU size.

Either the input or the output of a sequence of DLSDUs may be delayed excessively by the DLS users. Occurrences of delay caused by the DLS users are excluded in calculating average throughput values.

10.2.2 *Transit delay*

Transit delay is the elapsed time between DL-DATA request primitives and the corresponding DL-DATA indication primitives. Elapsed time values are calculated only on DLSDUs that are successfully transferred.

Succesful transfer of a DLSDU for the purposes of the QOS parameter is defined to occur when the DLSDU is transferred from the sending DLS user to he intended receiving DLS user without error, and in the proper sequence prior to release of the DLC by the receiving DLS user.

For connection-mode transfer transit delay is specified independently for each direction of transfer. Each specification is based on a previously stated average DLSDU size.

The transit delay for an individual DLSDU may be increased if the receiving DLS user exercises interface flow control. Such occurrences are excluded in calculating transit delay values.

10.2.3 *Residual Error Rate*

Residual error rate is the ratio of total incorrect, lost and duplicate DLSDUs to total DLSDUs transferred across the DLS boundary during a measurement period. The relationship among these quantities is defined, for a particular DLS user pair, as shown in Figure 3/X.212.

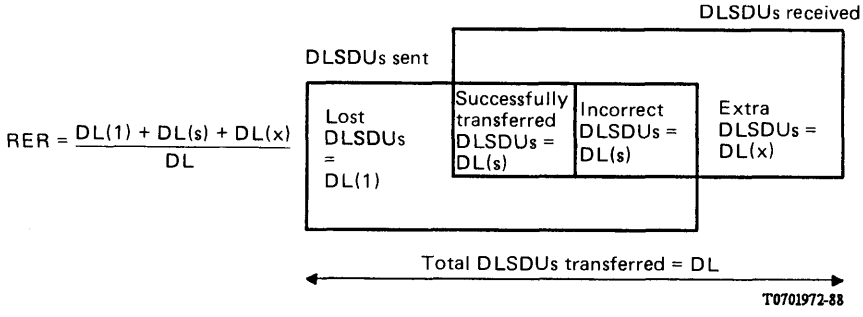


FIGURE 3/X.212

Components of Residual Error Rate

10.2.4 *Resilience*

This parameter specifies the probability of:

a) a DLS provider initiated DLC release (i.e. the isuance of a DL-DISCONNECT indication primitive with no prior DL-DISCONECT request primitive); or

b) a DLS provider initiated reset (i.e. the issuance of a DL-RESET indication primitive with no prior DL-RESET request primitive);

during a specified time interval on an established DLC.

10.2.5 *Protection*

Protection is the extent to which a DLS provider attempts to prevent unauthorized monitoring or manipulation of DLS user originated information. Protection is specified by a minimum and maximum protection option within a range of three possible protection options:

a) no protection features;

b) protection against passive monitoring; and

c) protection against modification, replay, addition or deletion.

Within the specified range, a DLS user selects a particular value during DLC establishment.

Each protection feature addresses a particular type of privacy or security threat, and each if available is typically provided by a different DLS provider mechanism.

10.2.6 *Priority*

The specification of priority is concerned with the relationship between DLCs.

This parameter specifies the relative importance of a DLC with respect to:

a) the order in which DLCs are to have their QOS degraded, if necessary; and

b) the order in which DLCs are to be released to recover resources, if necessary.

Priority is specified by a minimum and a maximum within a given range. Within the specified range, a DLS user selects a particular value during DLC establishment.

This parameter only has meaning in the context of some management entity of structure able to judge relative importance. The number of priority levels is limited.

11 Sequence of primitives

11.1 *Concepts used to define the connection-mode Data Link Service*

The service definition uses the following concepts:

a) DLC can be dynamically established or terminated between the DLS users for the purpose of exchanging data;

b) associated with each DLC, certain measures of QOS that are agreed between the DLS provider and the DLS users when the connection is established;

c) the DLC allows transmission of data and preserves its division into DLSDUs; the transmission of this data is subject to flow control;

d) the DLC can be returned to a defined state, and the activities of the two DLS users synchronized by the use of a Reset Service;

e) failure to provide the requested service may be signalled to the DLS user. There are three classes of failure:

   1) failures involving termination of the DLC;

   2) failures involving loss or duplication of user data, but without loss of DLC; and

   3) failures to provide the requested QOS without loss or duplication of user data or loss of the DLC.

11.2 *Constraints of Sequence of Primitives*

This section defines the constraints of the sequence in which the primitives defined in sections 12-14 may occur. The constraints determine the order in which primitives occur, but do not fully specify when they may occur. Other constraints, such as flow control of data, will affect the ability of a DLS user or a DLS provider to issue a primitive at any particular time.

The connection-mode primitives and their parameters are summarized in Table 4/X.212.

TABLE 4/X.212

Summary of data-link-connection-mode primitives and parameters

Phase	Service	Primitive	Parameters
DLC establishment	DLC establishment	DL-CONNECT request	(Called address, calling address, QOS parameter set)
		DL-CONNECT indication	(Called address, calling address, QOS parameter set)
		DL-CONNECT response	(Responding address, QOS parameter set)
		DL-CONNECT confirm	(Responding address, QOS parameter set)
Data transfer	Normal data transfer	DL-DATA request	(DLS user-data)
		DL-DATA indication	(DLS user-data)
	Reset	DL-RESET request	(Reason)
		DL-RESET indication	(Originator, reason)
		DL-RESET response	
		DL-RESET confirm	
DLC release	DLC release	DL-DISCONNECT request	(Reason)
		DL-DISCONNECT indication	(Originator reason)

11.2.1 *Relation of Primitives at the Two Endpoints*

A primitive issued at one DLC endoint will, in general, have consequences at the other DLC endpoint. The relations of primitives of each type at one DLC endpoint to primitives at the other DLC endpoint are defined in the appropriate subsections in sections 12-15; all these relations are summarized in the diagrams in Figure 4/X.212.

However, a DL-DISCONNECT request or indication primitive may terminate any of the other sequences before completion.

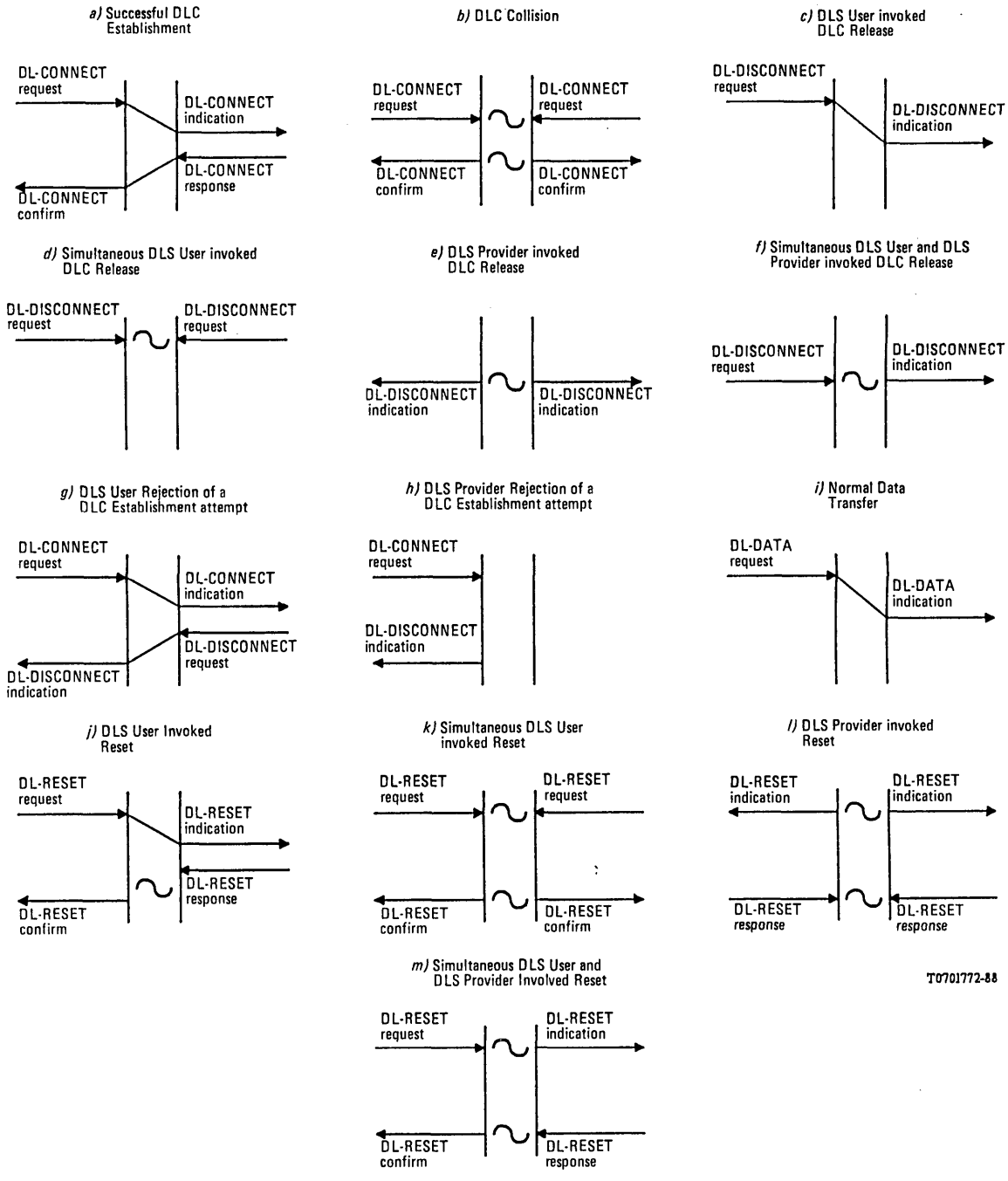


FIGURE 4/X.212

Summary of Data-link-connection-mode Service Primitive Time-sequence Diagram

11.2.2 *Sequence of primitives at one DLC endpoint*

The possible overall sequences of primitives at a DLC endoint are defined in the state transition diagram, Figure 5/X.212. In the diagram:

a) DL-DISCONNECT stands for either the request or the indication form of the primitive in all cases.

b) The labelling of the states "DLS user initiated reset pending" (5) and "DLS provider initiated reset pending" (6) indicate the party that started the local interaction, and does not necessarily reflect the value of the originator parameter.

c) The idle state (1) reflects the absence of a DLC. It is the initial and final state of any sequence, and once it has been re-entered, the DLC is released.

d) The use of a state transition diagram to describe the allowable sequences of service primitives does not impose any requirements or constraints on the internal organization of any implementations of the service.

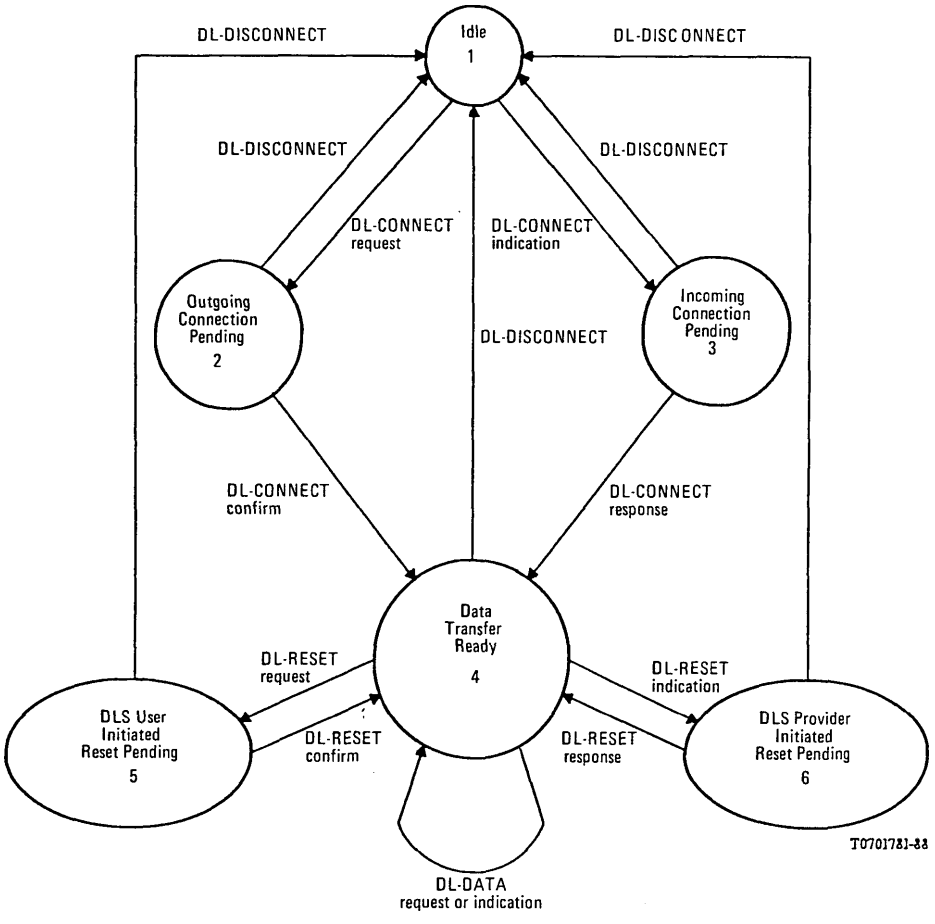


FIGURE 5/X.212

State transition diagram for Sequences of Data-link-connection-mode Service primitives at a DLC End-point

12 Connection establishment phase

12.1 *Function*

The connection establishment service primitives can be used to establish a DLC.

Simultaneous DL-CONNECT request primitives at the two DLSAPs result in one DLC as indicated in Figure 6/X.212.

12.2 *Types of primitives and parameters*

Table 5/X.212 indicates the types of primitives and the parameters needed for connection establishment.

TABLE 5/X.212

DLC establishment primitives and parameters

Parameter \ Primitive	DL-CONNECT request	DL-CONNECT indication	DL-CONNECT response	DL-CONNECT confirm
Called address	X	X(=) (see Note 2)		
Calling address	X (see Note 2)	X(=)		
Responding address			X (see Notes 1, 2)	X(=)
Quality of service parameter set	X	X	X	X

Note 1 – The need for responding address parameter if for further study.

Note 2 – This parameter may be implicitly associated with the DLSAP at which the primitive is issued.

12.2.1 *Addresses*

The parameters which take addresses as values (see §§ 12.2.2-12.2.4) all refer to DLSAP addresses.

Note – If the configuration allows any of these addresses to be known by the DL Entity on an *a priori* basis, then these DLSAP address(es) need not eplicitly be conveyed in the protocol.

12.2.2 *Called Address*

The called address parameter conveys an address identifying the DLSAP to which the DLC is to be established.

12.2.3 *Calling Address*

The calling address parameter conveys the address of the DLSAP from which the DLC has been requested.

12.2.4 *Responding Address*

The responding address parameter conveys the address of the DLSAP to which the DLC has been established.

12.2.5 *Quality of Service parameter set*

The use of the QOS parameter selection is not required when only one level of QOS is offered by the DLS provider.

12.2.5.1 *Throughput*

Two quality of service sub-parameters "target" and "lowest quality acceptable", which are in the agreed range, are passed to the DLS provider in the CONNECT request primitive. THE DLS provider will indicate to the DLS users, the "available" throughput in the DL-CONNECT confirm, and DL-CONNECT indication. The "available" parameter shall be a value from the range between the "target" and "lowest quality acceptable" (see § 10.2.1).

12.2.5.2 *Selected protection*

The parameter specifies a particular degree of protection, within the agreed range (see § 10.2.5) for the DLSDU of any subsequently submitted DL-DATA request primitive transferred on the DLC.

12.2.5.3 *Selected priority*

The parameter specifies a particular priority, within the agreed range (see § 10.2.6), for the DLSDU of any subsequent DL-DATA request primitive transferred on the DLC.

12.3 *Sequence of primitives*

The sequence of primitives in a successful connection establishment is defined by the time-sequence diagram in Figure 6/X.212.

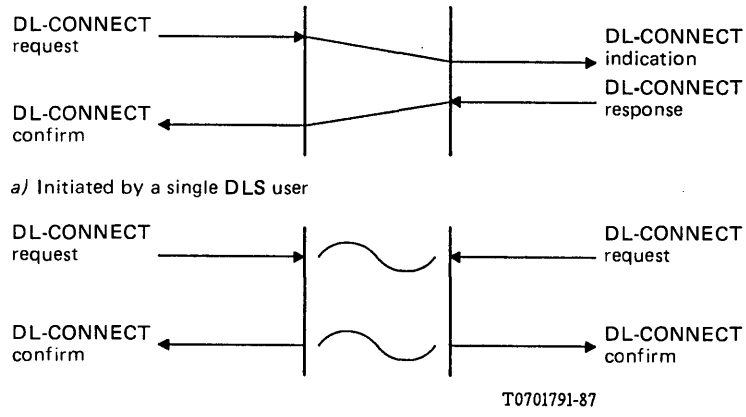


FIGURE 6/X.212

DLC Establishment

These DLC establishment procedures may fail either due to the inability of the DLS provider to establish a DLC or due to the unwillingness of the called DLS user to accept a DL-CONNECT indication primitive (for these cases see DLC release service, §§ 13.4 and 13.5).

13 Connection release phase

13.1 *Function*

The connection release service primitives are used to release a DLC. The release may be initiated by any of the following:

a) either or both of the DLS users, to release an established DLC;

b) the DLS providers to release an established DLC; all failures to maintain a DLC are indicated in this way;

c) the DLS user, to reject a DL-CONNECT indication;

d) the DLS provider, to indicate its inability to establish a requested DLC; or

e) the DLS user which sent the DL-CONNECT request, to abandon the connection attempt before the connection has been made available for use by receipt of a DL-CONNECT primitive.

Initiation of the release service element is permitted at any time regardless of the current phase of the DLC. Once a release service has been initiated, the DLC will be disconnected. A DL-DISCONNECT request cannot be rejected. The DLS does not guarantee delivery of any DLSDU associated with the DLC once the release phase is entered.

13.2 *Types of primitive and parameters*

Table 6/X.212 indicated the types of primitives and the parameters needed for connection release.

TABLE 6/X.212

DLC release primitives and parameters

Parameter \ Primitive	DL-DISCONNECT request	DL-DISCONNECT indication
Originator		X
Reason	X	

13.2.1 *Originator*

The originator parameter indicates the source of the release. Its value indicates either the DLS user, the DLS provider, or that the originator is unkown.

13.2.2 *Reason*

The reason parameter gives information about the cause of the release. The value conveyed in this parameter will be as follows:

a) when the originator parameter indicates a DLS provider generated release, the value is one of:
   1) "disconnection-permanent condition";
   2) "disconnection-transient condition";
   3) "connection rejection-DLSAP-address unknown";
   4) "connection rejection-DLSAP unreachable/permanent condition";
   5) "connection rejection-DLSAP unreachable/transient condition";
   6) "connection rejection-QOS not available/permanent condition";
   7) "connection rejection-QOS not available/transient condition"; or
   8) "reason unspecified";

   Note – Addition to, or refinement of this list of values to convey more specific diagnostic and management information is for further study.

b) when the originator parameter indicates DLS user initiated release, the value is one of:
   1) "disconnection-normal condition";
   2) "disconnection-abnormal condition";
   3) "connection rejection-permanent condition";
   4) "connection rejection-transient condition"; or
   5) "reason unspecified"; and

c) when the originator parameter indicates an unknown originator the value of the Reason parameter is "reason unspecified". This allows the parameters to be implied when they cannot be explicitly conveyed in the Data Link protocol.

13.3 *Sequence of primitives when releasing an established DLC*

The sequence of primitives depends on the origins of the release action. The sequence may be:

a) initiated by one DLS user, with a request from that DLS user leading to an indication to the other;

b) initiated by both DSL users, with a request from each of the DLS users;

c) initiated by the DLS provider, with an indication to each of the DLS users; or

d) initiated independently by one DLS user and the DLS provider, with a request from the originating DLS user and an indication to the other.

The sequences of primitives in these four cases are defined by the time-sequence diagrams in Figures 7/X.212-10/X.212.

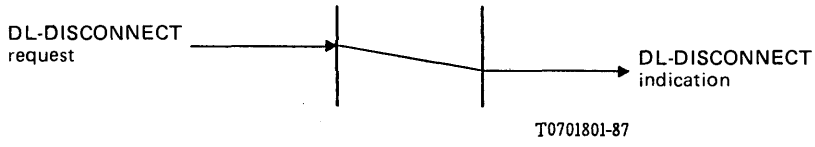


FIGURE 7/X.212

DLS User invocation

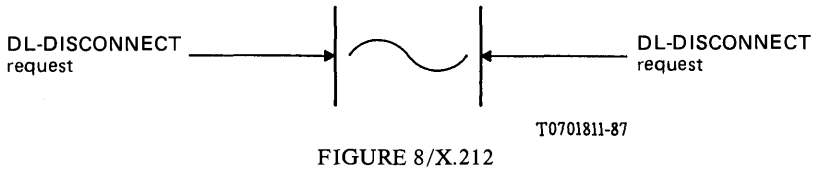


FIGURE 8/X.212

Simultaneous invocation by both DLS Users

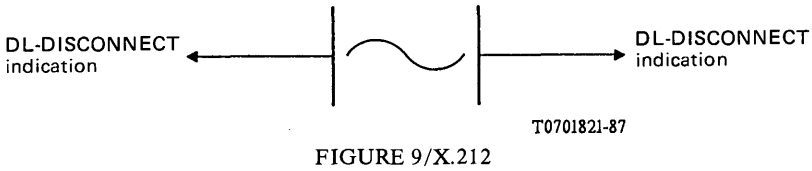


FIGURE 9/X.212

DLS Provider invocation

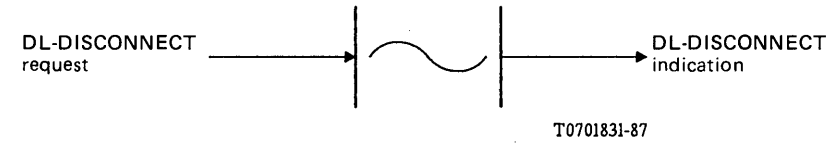


FIGURE 10/X.212

Simultaneous DLS User and DLS Provider invocations

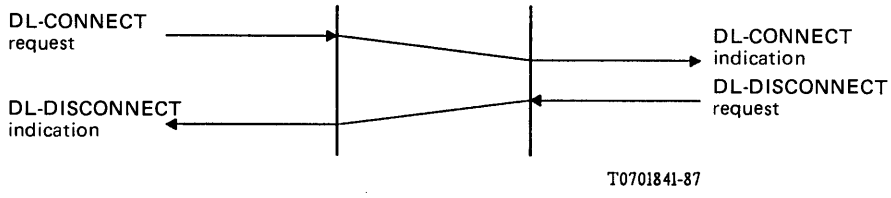


FIGURE 11/X.212

Sequence of primitives in DLS User rejection of a DLC connection attempt

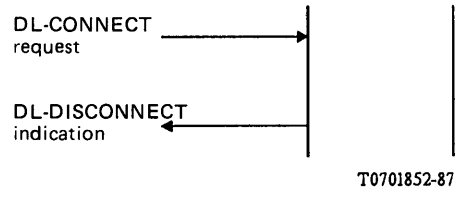


FIGURE 12/X.212

Sequence of primitive in DLS Provider rejection of a DLC establishment attempt

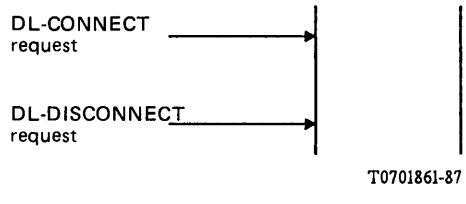


FIGURE 13/X.212

Both primitives are destroyed in the queue

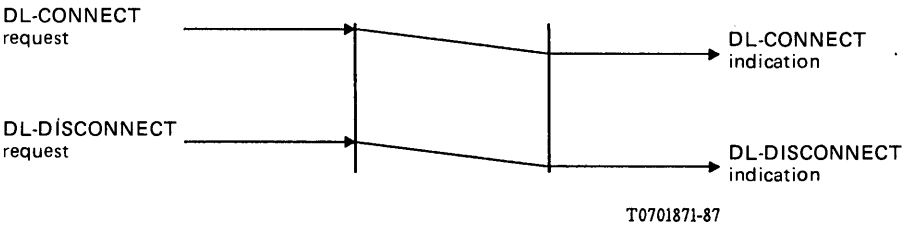


FIGURE 14/X.212

DL-DISCONNECT indication arrives before DL-CONNECT response is sent

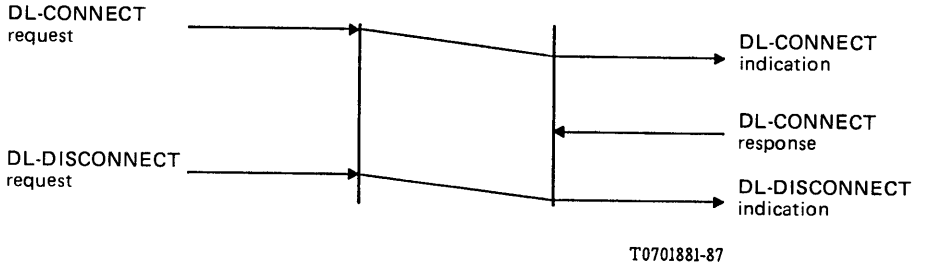


FIGURE 15/X.212

DL-DISCONNECT indication arrives after DL-CONNECT response is sent

13.4 *Sequence of primitives in a DLS user rejection of DLC establishment attempt*

A DLS user may reject a DLC establishment attempt by using a DL-DISCONNECT request. The originator parameter in the DL-DISCONNECT primitives will indicate DLS user initiated release. The sequence of events is defined in the time-sequence diagram in Figure 11/X.212.

13.5 *Sequence of primitives in a DLS provider rejection of a DLC establishment attempt*

If the DLS provider is unable to establish a DLC, it indicates this to the requester by a DL-DISCONNECT indication. The originator parameter in this primitive indicates a DLS provider originated release. The sequence of events is defined in the time-sequence diagram in Figure 12/X.212.

13.6 *Sequence of Primitives in a DLS User Abort of a DLC Establishment Attempt*

If the DLS user, having previously sent a DL-CONNECT request and not received a DL-CONNECT confirm or DL-DISCONNECT indication, wishes to abort the DLC establishment attempts the DLS user shall issue a DL-DISCONNECT request. The resulting sequence of primitives is dependent upon the relative timing of the primitives involved and the transit delay of the DLS provider as defined by the time-sequence diagrams in Figures 13 to 15/X.212. No information can be implied by detecting which of these alternatives occur.

14 Data transfer phase

14.1 *Data transfer*

14.1.1 *Function*

The data transfer service primitives provide for an exchange of user-data (DLSDUs), in either direction or in both directions simultaneously on a DLC. The DLS preserves both the sequence and the boundaries of the DLSDUs.

Note – Designers of protocols using DLS should realize that the requested QOS applies to complete DLSDUs, and that divisions of available data into small DLSDUs may have cost implications because of the impact on cost optimization mechanisms operated by the DLS provider.

14.1.2 *Types of primitives and parameter*

Table 7/X.212 indicates the types of primitives and the parameters needed for data transfer.

TABLE 7/X.212

Data transfer primitives and parameter

Parameter \ Primitive	DL-DATA request	DL-DATA indication
DLS user-data	X	X(=)

14.1.2.1 *DLS user-data*

This parameter allows the transmission of DLS user-data between DLS users, without modification by the DLS provider. The DLS user may transmit any integral number of octets greater than zero up to a limit determined by the DLS provider. The value of this limit is made available to the DLS user by the use of management facilities or *a priori* knowledge.

14.1.3 *Sequence of primitives*

The operation of the DLS in transferring DLSDUs can be modelled as a queue of unknown size within the DLS provider (see § 9). The ability of a DLS user to issue a DL-DATA request or of the DLS provider to issue a DL-DATA indication depends on the behaviour of the receiving DLS user and the resulting state of the queue.

The sequence of primitives in a successful data transfer is defined in the time-sequence diagram in Figure 16/X.212.

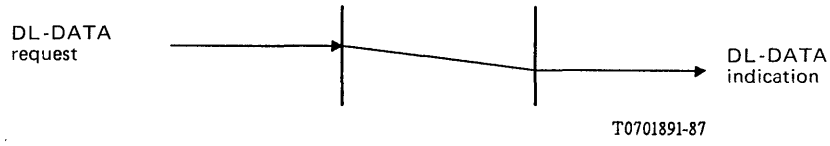


FIGURE 16/X.212

Sequence of primitives for normal data transfer service

The above sequence of primitives may remain uncompleted if a DL-RESET or a DL-DISCONNECT primitive occurs.

14.2 *Reset service*

14.2.1 *Function*

The Reset service may be used:

a) by the DLS user, to resynchronize the use of the DLC; or

b) by the DLS provider, to report detected loss of data unrecoverable within the DLS. All loss of data which does not involve loss of the DLC is reported in this way.

Invocation of the Reset service will unblock the flow of DLSDUs in case of congestion of the DLC; it will cause the DLS provider to discard DLSDUs, and to notify user or users that did not invoke Reset that a Reset has occurred. The service will be completed in a finite time, irrespective of the acceptance of DLSDUs. Any DLSDUs not delivered to the DLS users before completion of the service will be discarded by the DLS provider.

Note – A Reset may require a recovery procedure to be performed by the DLS users.

14.2.2 *Types of primitives and parameters*

Table 8/X.212 indicates the types of primitives and the parameters needed for the reset service.

TABLE 8/X.212

Reset primitives and parameters

Primitive / Parameter	DL-RESET request	DL-RESET indication	DL-RESET response	DL-RESET confirm
Originator		X		
Reason	X	X		

14.2.2.1 *Originator*

The originator parameter indicates the source of the Reset. Its value indicates either the DLS user, the DLS provider, or that the originator is unknown.

14.2.2.2 *Reason*

The reason parameters give information indicating the cause of the Reset. The value conveyed in this parameter will be as follows:

a) when the originator parameter indicates a DLS provider generated Reset, the value is one of:

1) "Data Link flow control congestion"; or

2) "Data Link error";

Note – Addition to or refinement of this list of values to convey more specific diagnostic or management information is for further study.

b) when the originator parameter indicates a DLS user initiated Reset, the value is "user resynchronization"; and

c) when the originator parameter indicates an unknown originator, the value is "reason unspecified". This allows the parameters to be implied when they cannot be explicitly conveyed in the Data Link protocol.

14.2.3 *Sequence of primitives*

The interaction between each DLS user and the DLS provider shall be either one of the following exchanges of primitives:

a) a DL-RESET request from the DLS user, followed by a DL-RESET confirm from the DLS provider; or

b) a DL-RESET indication from the DLS provider, followed by a DL-RESET response from the DLS user.

The DL-RESET request acts as a synchronization mark in the stream of DLSDUs that are transmitted by the issuing DLS user; the DL-RESET indication likewise acts as a synchronization mark in the stream of DLSDUs by the peer DLS user. Similarly, the DL-RESET response acts as a synchronizing mark in the stream of DLSDUs transmitted by the responding DLS user, while the DL-RESET confirmation acts as a synchronization mark in the stream of DLSDUs that are received by the DLS user which originally issued the reset.

The resynchronization properties of the Reset Service are that:

1) No DLSDU transmitted by the DLS user *before* the synchronization mark in that transmitted stream will be delivered to the other DLS user *after* the synchronization mark in that received stream.

   The DLS provider will discard all DLSDUs, submitted before the issuing of the DL-RESET request that have not been delivered to the peer DLS user when the DLS provider issues the DL-RESET indication.

   Also, the DLS provider will discard all DLSDUs, submitted before the issuing of the DL-RESET that have not been delivered to the initiator of the DL-RESET when the DLS provider issues the DL-RESET confirm.

2) No DLSDU transmitted by a DLS user *after* the synchronization mark in that transmitted stream will be delivered to the other DLS user *before* the synchronization mark in that received stream.

The complete sequence of primitives depends upon the origin of the Reset action and the occurrence or otherwise of conflicting origins. Thus the Reset Service may be:

i) invoked by one DLS user, leading to interaction a) with that DLS user and interaction b) with the peer DLS user;

ii) invoked by both DLS users, leading to interaction a) with both DLS users;

iii) invoked by the DLS provider, leading to interaction b) with both DLS users; or

iv) invoked by one DLS user and DLS provider, leading to interaction a) with the originating DLS user and b) with the peer DLS user.

The sequence of primitives in these four cases is defined in the time-sequence diagrams in Figures 17-20/X.212.

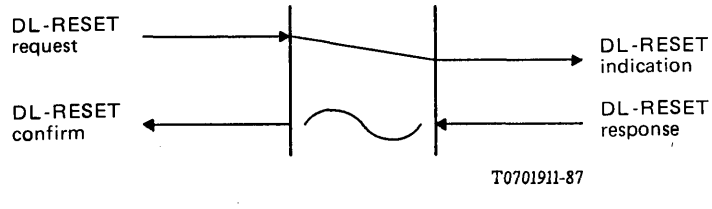


FIGURE 17/X.212

Sequence of primitives in a DLS user initiated reset service

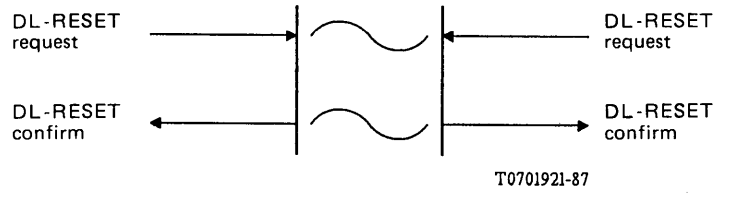


FIGURE 18/X.212

Sequence of primitives in a simultaneous DLS user initiated reset service

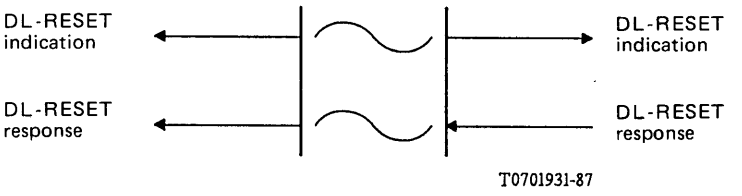


FIGURE 19/X.212

Sequence of primitives in a DLS provider initiated reset service

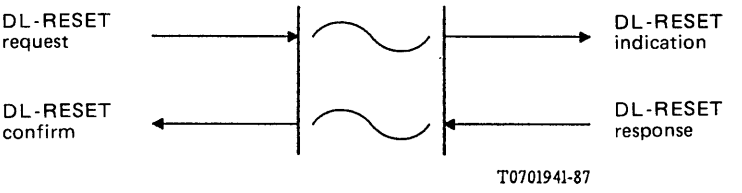


FIGURE 20/X.212

Sequence of primitives in a simultaneous DLS user and DLS provider initiated reset service

The above sequences of primitives may remain uncompleted if a DL-DISCONNECT primitive occurs.

PART THREE – DEFINITION OF THE CONNECTIONLESS-MODE SERVICE

15 Features of the Connectionless-mode Data Link Service

The DLS provides the following features to the DLS user:

a) a means by which DLSDUs of limited length are delimited and transparently transmitted from one source DLSAP to a destination DLSAP in a single DLS access, without establishing or later releasing a DLC;

b) associated with each instance of connectionless-mode transmission, certain measures of QOS which are selected by the sending DLS user when the connectionless-mode transmission is initiated.

16 Model of the Connectionless-mode Data Link Service

This Recommendation uses the abstract model for a layer service defined in § 4 of Recommendation X.210. The model defines the interactions between the DLS users and DLS provider which takes place at the two DLSAPs. Information is passed between the DLS user and the DLS provider by service primitives, which may convey parameters.

16.1 *Model of a Data-link-connectionless-mode Data Transmission*

A defining characteristic of data-link-connectionless-mode data transmission is the independent nature of each invocation of the DL-connectionless-mode service (see Appendix II).

In practice, however, it is often possible to relate to DLS users certain characteristics of the service for an association existing between a given pair of DLSAPs, which enhance the basic data-link-connectionless-mode service in order to effectively correlate the choice of Network Layer protocol type with the service provided.

Note – It is anticipated that such information is made available to the DLS user through some management facility (or set of facilities).

Thus as a descriptive aid the data-link-connectionless-mode service – as provided between any two DLSAPs – can be modelled in the abstract as an association between the two DLSAPs. This association is permanent.

Only one type of object, the unitdata object, can be handed over to the DLS provider via a DLSAP. In Figure 21/X.212, DLS user A represents the DLS user that passes objects to the DLS provider. DLS user B represents the DLS user that accepts objects from the DLS provider.

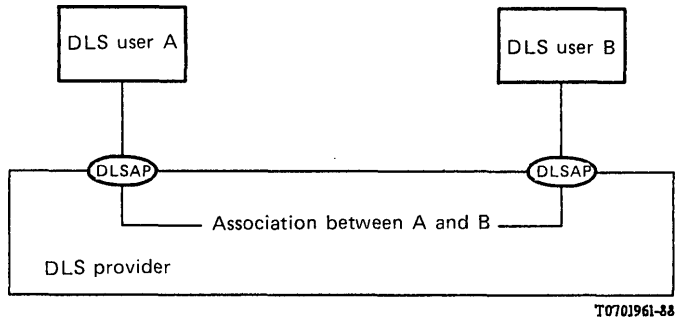


FIGURE 21/X.212

Mod ' for a connectionless DLS data transmission

In general, the DLS provider may perform any or all of the following actions:

a) discard objects;

b) duplicate objects; and/or

c) change the order of service request into a different order of service indications.

However, with respect to a given association, some characteristics of the nature and type of service, beyond those attributed to the basic DL-connectionless-mode service, may be relatd to the DLS user through some management facility. The following are examples of some requirements or constraints that may be assumed/observed by the DLS user:

a) objects will not be discarded;

b) objects will not be duplicated; and

c) the order of the service indications will be the same as the order of the service requests.

Where such information is made known to the DLS user prior to the invocation of the DL-connectionless-mode service, it may make use of such knowledge to select an appropriate Network Layer protocol.

The operations that are performed by the DLS provider for a particular DL association do not depend on the behaviour of the DLS users. Awareness of the characteristics of the DLS provided is part of the DLS users' *a priori* knowledge of the OSI environment.

17 Quality of Connectionless-mode Service

The term "Quality of Service" (QOS) refers to certain characteristics of a connectionless-mode data transmission as observed between the DLSAPs. QOS describes aspects of a connectionless-mode data transmission which are solely attributable to the DLS provider; it can only be properly determined in the absence of DLS user behaviour (which is beyond the control of the DLS provider) that specifically constrains or impedes the performance of the DLS.

Whether the view of the QOS during each instance of the use of connectionless-mode data transmission is the same to each DLS user associated with the service depends on the nature of their association and the type of information concerning the nature of the service made available to the DLS user(s) by the DLS provider prior to the invocation of the service.

17.1 *Determination of QOS for Connectionless-mode Service*

A basic characteristic of a connectionless-mode service is that, unlike a connection-mode service, no dynamic association similar to that during a connection establishment is set up between the parties involved. Thus, the service characteristics to be provided during the transfer are not selected on a per DLC basis.

Associated with each DL connectionless-mode transmission, certain measures of QOS are requested by the sending DLS user when the primitive action is initiated. The requested measures (or parameter values) and options, are based on *a priori* knowledge by the DLS user of the service(s) made available to it by the DLS provider. Knowledge of the characteristics and type of service provided (i.e., the parameters, formats, and options that affect the transfer of data) is made available to a DLS user through some layer management interaction prior to (any) invocation of the DL-connectionless-mode service. Thus, the DLS user not only has knowledge of the parties with which it may communicate, it also has explicit knowledge of the characteristics of the service it can expect to be provided with each invocation of the service.

The DLS provider may also provide information on the current QOS independently of access to the service by a DLS user. This seemingly dynamic aspect of QOS determination is not a negotiation but provided with knowledge of the characteristics of the service currently outside of any instance of the invocation of the service.

17.2 *Definition of connectionless-mode QOS parameters*

QOS parameters are classified as:

a) parameters that express DLS performance, as shown in Table 9/X.212.

TABLE 9/X.212

Classification of performance QOS-parameters

Performance criterion	
Speed	Accuracy/reliability
Transit delay	Residual error rate (corruption, duplication/loss)

b) parameters that express other DLS characteristics, as shown in Table 10/X.212.

TABLE 10/X.212

QOS-parameters not associated with performance

Protection
Priority

Note – Some QOS-parameters are defined in terms of the issuance of DLS primitives. Reference to a DLS primitive refers to the complete execution of that DLS primitive at the appropriate DLSAP.

17.2.1 *Transit delay*

Transit delay is the elapsed time between DL-UNITDATA request primitives and the corresponding DL-UNITDATA indication primitives. Elapsed time values are calculated only on DLDSUs that are successfully transferred.

Successful transfer of a DLSDU is defined, for the purpose of this QOS parameter, to occur when the DLSDU is transferred from the sending DLS user to the intended receiving DLS user without error.

For connectionless-mode transfer, transit delay is specified independently for each Data-link-connection-less-mode data transmission.

The transit delay for an individual DLSDU may be increased if the receiving DLS user exercises interface flow control. Such occurrences are excluded in calculating both average and maximum transit delay values.

17.2.2 *Residual Error Rate*

Residual error rate is the ratio of total incorrect, lost and duplicate DLSDUs to total DLSDUs transferred across the DLS boundary during a measurement period. The relationship among these quantities is defined, for a particular DLS user pair, as shown in Figure 22/X.212.

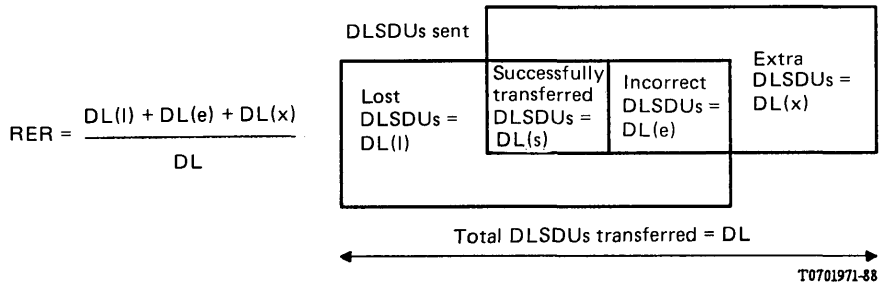


FIGURE 22/X.212

Components of Residual Error Rate

17.2.3 *Protection*

Protection is the extent to which a DLS provider attempts to prevent unauthorized monitoring or manipulation of DLS user originated information. Protection is specified by a minimum and maximum protection option within a range of three possible protection options:

a) no protection features;

b) protection against passive monitoring; and

c) protection against modification, replay, addition or deletion.

Within the specified range, a DLS user selects a particular value for each DLSDU submitted or connectionless-mode data transmission.

Each protection feature addresses a particular type of privacy or security threat and each is typically provided by a different DLS provider mechanism.

17.2.4 *Priority*

The specification of priority is concerned with the relationship between connectionless-mode data transfer invocations.

This parameter specifies the relative importance of unidata objects with respect to gaining use of shared resources.

This parameter only has meaning in the context of some management entity of structure able to judge relative importance. The number of priority levels is limited.

18 Sequence of connectionless-mode primitives at one DLSAP

The possible overall allowed sequence of primitives at a DLSAP are defined in the state transition diagram in Figure 23/X.212.

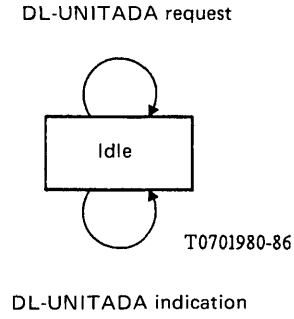


FIGURE 23/X.212

State transition diagram for sequences of connectionless-mode primitives at one DLSAP

19 Data transfer

19.1 *Function*

Data-Link connectionless-mode data transmission service primitives can be used to transmit an independent, self-contained DLSDU from one DLSAP to another DLSAP in a single DL service access. The DLSDU is independent in the sense that it bears no relationship to any other DLSDU transmitted through the invocation of the connectionless-mode service or the connection-mode service (unless specific QOS requests have been accepted). It is self-contained in that all of the information required to deliver the DLSDU is presented to the DLS provider, together with the user data to be transmitted, in a single service access; thus, no initial establishment or subsequent release of a DLS is required, provided that the DLS users exist and are known to the DLS provider.

A DLSDU transmitted using DL connectionless-mode data transmission is not considred by the DLS provider to be related in any way to any other DLSDU. Although the DLS maintains the integrity of individual DLSDUs, it does not necessarily deliver them to the receiving DLS user in the order in which they are presented by the sending DLS user.

No means are provided by which the receiving DLS user may control the rate at which the sending DLS user may send DLSDU (peer-to-peer flow control). The DLS provider will not maintain any state information relative to any aspects of the flow of information between any specific combination of DLSAPs. Flow control exerted by the DLS provider upon the sending DLS user can only be described in terms of a specific interface.

19.2 *Types of primitives and parameters*

Table 11/X.212 indicates the types of primitives and parameters needed for the Data-link-connectionless-mode data transmission service.

TABLE 11/X.212

DL-connectionless-mode data transfer primitives and parameters

Parameter \ Primitive	DL-UNIDATA request	DL-UNIDATA indication
Source address	X	X(=)
Destination address	X	X(=)
QOS-parameter set	X	X (see Note)
DLS user-data	X	X(=)

Note – The need for QOS-parameters to be included in the DL-UNIDATA indication is for further study.

19.2.1 *Addresses*

The addresses referred to in Table 11/X.212 are DLSAP addresses. The co..nection-mode and connectionless-mode DLSs may both use the same DLSAP addresses.

Note – If the configuration allows any of these addresses to be known by the DL Entity on an *a priori* basis, then these DLSAP addresses need not explicitly be conveyed in this protocol.

19.2.2 *Quality of Service*

The value of the QOS parameter is a list of sub-parameters. For each parameter, the values on the two primitives are related so that:

a) on the request primitive, any defined value is allowed; and

b) on the indication primitive, the quality of service indicated is less than or equal to the value specified for the corresponding request primitive.

The use of the QOS parameter selection is not required when only one level of QOS is offered by the DLS provider.

19.2.3 *DLS User-Data*

This parameter allows the transmission of DLS user-data between DLS users, without modification by the DLS provider. The DLS user may transmit any integral number of octets greater than zero up to a limit determined by the DLS provider. The value of this limit is made available to the DLS user by the use of management facilities or *a priori* knowledge.

19.3 *Sequence of primitives*

The sequence of primitives in a successful DL connectionless-mode data transmission is defined in the time-sequence diagram in Figure 24/X.212.

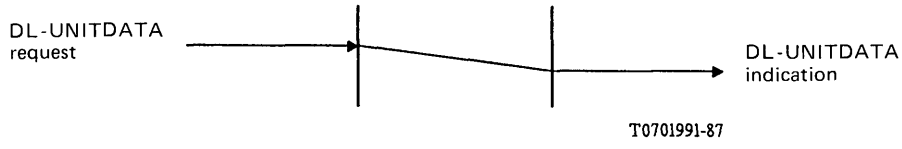


FIGURE 24/X.212

Sequence of primitives in connectionless-mode data transfer

APPENDIX I

(to Recommendation X.212)

Differences between CCITT and ISO texts

The following differences between this Recommendation and ISO 8886, Information Processing Systems – Data Communications – Data Link Service Definition, should be noted.

1. Appendix II to this Recommendation, which describes the relationship between connection-mode and connectionless-mode operation, is not present in ISO 8886.

2. Appendix III to this Recommendation, which defines a method for providing the OSI connection-mode Data Link Service using X.25 LAPB compatible DTE procedures, is not present in ISO 8886.

APPENDIX II

(to Recommendation X.212)

The relationship between the two types of Data Link Service

This appendix does not form a part of this Recommendation.

II.1 *Introduction*

Recommendation X.200 describes the Reference Model of Open Systems Interconnection for CCITT Applications. It is the intention of Recommendation X.200 that the Reference Model should establish a framework for coordinating the development of existing and future Recommendations for interconnection of systems.

The assumption that a connection is a fundamental prerequisite for communication in an OSI environment permeates the Reference Model and is one of the most useful and important unifying concepts of the OSI architecture. Further study is under way to determine whether it is appropriate to include provisions for a connectionless-mode operation to expand the scope of application of the OSI Reference Model of Recommendation X.200.

This appendix has been produced to introduce the terms and concepts of the connectionless-mode of operation to facilitate the study of this provision. The two alternatives (connection-mode transmission and connectionless-mode transmission) can be treated as complementary concepts and can be applied appropriately in the different circumstances for which each is suited.

II.2 *What is connectionless-mode transmission*

Connectionless-mode transmission in one form or another has been a consideration in the specification of services and protocols for data communication. The terms "message mode", "datagram", "transaction mode" and "connection free" have been used in the literature to describe variations on the same basic theme; the transmission of a unit of data in a single self-contained operation without establishing, maintaining and releasing a connection.

Since connectionless-mode transmission and connection-mode transmission are complementary concepts, they are best understood in juxtaposition, particularly since connectionless-mode transmission is defined most easily with respect to its relationship to the concept of a connection.

II.2.1 *Connection-mode transmission*

In the formal terminology of the Reference Model, a connection is an association established for the transfer of data between two or more peer-entities. This association is established between the peer-entities themselves and between each entity and the next lower layer. The ability to establish a connection and to transfer data over it is provided to the entities in a given layer by the next lower layer as a connection-mode service. An instance of the use of a connection-mode service by peer-entities proceeds through three distinct phases:

a) connection establishment,

b) data transfer,

c) connection release.

In addition to the clearly distinguishable lifetime exhibited by these phases, a connection has the following fundamental characteristics:

i) it involves establishing and maintaining a three or more party agreement concerning the transfer of data between the peer-entities concerned and the layer providing the service;

ii) it allows the negotiation among all parties concerned of the parameters and options that will govern the transfer of data;

iii) it provides connection identification by means of which the overheads associated with address resolution and address transmission can be avoided during the data transfer phase;

iv) it provides a context within which successive units of data transferred between the peer-entities are logically related, and therefore makes it possible to maintain sequence and provide flow control.

The characteristics of connection-mode transmission are attractive in applications that call for relatively long-lived, stream-oriented interactions between entities in stable configurations. In these cases, the entities involved:

1) initially discuss their requirements and agree to the terms of their interaction, reserving whatever resources they may need;

2) transfer a series of related units of data to accomplish their mutual objective; and

3) explicitly end their interaction, releasing the previously reserved resources.

The properties of connection-mode transmission are also relevant in a wide range of other applications.

II.2.2 *Connectionless-mode transmission*

In formal terminology, connectionless-mode transmission is the transmission of a single unit of data from a source service-access-point to one or more destination service-access-points without establishing a connection. A connectionless-mode transmission service allows an entity to initiate such a transmission by the performance of a single service access.

In contrast to a connection, an instance of the use of the connectionless-mode service does not have a clearly distinguishable lifetime. In addition, the connectionless service, unless otherwise explicitly determined, has the following fundamental characteristics:

a) it requires only a pre-existing association between the peer-entities involved, which determines the characteristics of the data to be transmitted, and a two-party agreement between each peer-entity and the service provider; no dynamic peer-to-peer agreement is involved in an instance of the use of the service;

b) all of the information required to deliver a unit of data (destination address, quality of service selection, service options, etc.), is presented to the layer providing the connectionless-mode service, together with the user data to be transmitted, in a single access that is not required to relate to any other service access;

c) each unit of data transmitted is entirely self-contained and can be routed independently by the service provider.

Connectionless-mode transmission creates no relationship, expressed or implied, between data units. Nothing about the service invocation, the transmission of the data by the connectionless-mode service, or the data unit itself, affects or is affected by any other past, present or future operation, whether connection-mode or connectionless-mode. A series of data units handed one after the other to a connectionless-mode service for delivery to the same destination will not necessarily be delivered to the destination in the same order. The connectionless-mode service will not necessarily report or recover instances of non-delivery.

In practice, management facilities may be utilized in order to convey characteristics of the nature, quality and type of service provided by one layer to the next higher layer of the architecture prior to the invocation of that service. This facility (or set of facilities) may provide information concerning the aforementioned service characteristics in an *a priori* manner, thereby providing information which may be relied upon for subsequent invocations of the service. In contrast, the information may be provided in a dynamic fashion, invoked (perhaps) prior to each instance of use of the connectionless-mode service. Where added value may be determined prior to the use of a connectionless-mode service, parameters relating such characteristics will be related in a complementary fashion to the request for provision of the service.

II.3 *General architectural principles*

II.3.1 *Basic concepts*

In order for (N+1)-entities to communicate using either an (N)-connection-mode service or an (N)-connectionless-mode service, there must be a pre-arranged association between them constituted by the *a priori* knowledge that each must have of the other(s) in order at least to initiate the use of the service. The association comprises four elements:

a) knowledge of the addresses of the peer-entities involved;

b) knowledge of a protocol agreed by the peer-entities for use at least for initial communication;

c) knowledge of the probable availability for communication of the peer-entities;

d) knowledge of the Quality of Service on offer from the (N)-service.

II.3.2 *Communication between peer-entities*

(N + 1)-entities can communicate using an (N)-connectionless-mode service provided that there is a pre-arranged association between them providing knowledge about each other that allows them to do so. This knowledge must allow the location of (N + 1)-entities to be determined, and it must determine the correct interpretation of (N) service data units by a receiving (N + 1)-entity. It may define the rates of transfer, rates of response, and the protocol in use between the (N + 1)-entities. The knowledge may result from a prior agreement between the (N + 1)-entities on the parameters, formats and options to be used.

There are instances where the connectionless-mode service provided by the (N + 1)-layer does not provide direct access between (N)-service-access-points supported by the layer. Connectionless-mode transmission can still occur between these service-access-points if one or more (N)-entities provide a relay. The fact that an (N + 1)-connectionless-mode transmission is relayed by one or more (N)-entities is known neither by the (N − 1) layer nor the (N + 1)-layer.

APPENDIX III

(to Recommendation X.212)

Use of the X.25 LAPB-compatible DTE data link procedures to provide the connection-mode Data Link Service for Open Systems Interconnection for CCITT applications

This appendix defines a method for providing the OSI Connection-mode Data Link Service (CODLS) through the use of the X.25 LAPB-compatible DTE data link procedures as described in X.25 and X.75 (abbreviated to X.25/LAPB, for the remainder of this document).

The relationship between the X.25 LAPB-compatible DTE data link procedures and the OSI CODLS is shown in Figure III-1/X.212. This relationship is described only in terms of the Data Link Layer entities that provide the CDLS.

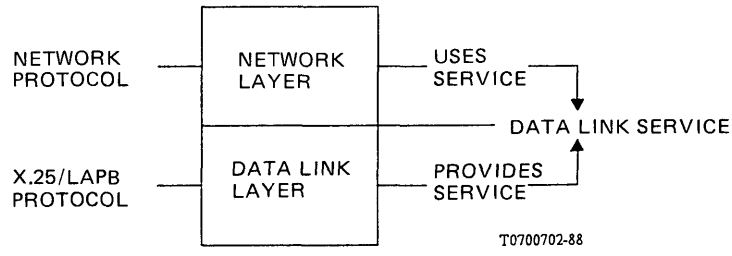


FIGURE III-1/X.212

Relationship of the X.25 LAPB to the OSI Connection-mode Data Link Service

III.1 *Scope and field of application*

The OSI CODLS, 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 CODLS and the real elements of the protocol. This appendix provides such a mapping for the X.25/LAPB single link procedure (the extension of this document to multi-link procedures and other layer 2 protocols is for further study).

This appendix specifies a set of requirements applicable both to end systems and to the operation of half of an intermediate system (i.e. the Data Link Layer of an Interworking Unit/Relay Open System which operates relaying at the network layer).

III.2 *References*

In addition to the references identified in § 2 of this Recommendation, the following references are applicable to this appendix.

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.
Recommendation X.75	Packet-switched signalling system between public networks providing data transmission services.
ISO 7776	Information processing systems – Data communications – High level data link control procedure – Description of the X.25 LAPB-compatible DTE data link procedures.

III.3 *Definitions*

III.3.1 *Reference model definitions*

The following terms, as defined in Recommendation X.200, are used in this appendix.

- Data Link Connection;
- Data Link Layer;
- Data Link Service;
- Data Link Service Access Point;
- Data Link Service Access Point address.

III.3.2 *Service convention definitions*

The following terms, as defined in Recommendation X.210, are used in this appendix:

- Data Link Service user;
- Data Link Service provider;
- primitive;
- request;
- indication;
- response;
- confirm.

III.3.3 *X.25 definitions*

The following terms, as defined in Recommendation X.25, are used in this appendix:

- Data Link;
- Data Terminal Equipment;
- Data Circuit-Terminating Equipment.

III.4 *Abbreviations*

III.4.1 *Data Link Service abbreviations*

CODLS	Connection-mode Data Link Service
DLL	Data Link Layer
DLSAP	Data Link Service Access Point
OSI	Open Systems Interconnection
QOS	Quality of Service
DLC	Data Link Connection
DLS	Data Link Service

III.4.2 *X.25/LAPB abbreviations*

LAPB	Link Access Procedure Balanced mode
I	Information (frame)
DM	Disconnected Mode (frame)
SABM	Set Asynchronous Balanced Mode (frame)
UA	Unnumbered Acknowledgement (frame)
FRMR	Frame Reject (frame)
RR	Receive Ready (frame)
RNR	Receive Not Ready (frame)
REJ	Reject (frame)
DTE	Data Terminal Equipment
DCE	Data Circuit-terminating Equipment
DXE	either a DTE or a DCE
SABME	Set Asynchronous Balanced Mode Extended

III.5 *Overview*

The Data Link Service provides for the transparent transfer of the data between the DLS users.

III.5.1 *Elements of the X.25/LAPB used to support the OSI CODLS*

The X.25/LAPB, as defined in X.25 and X.75, provides a specific realization for the transparent transfer of the data between DLS users of the CODLS. The elements of this protocol to be considered are the frames and the fields to be mapped to the primitives and parameters of the OSI CODLS.

Table III-1/X.212 below lists the X.25/LAPB frames and associated fields used when supporting the OSI CODLS.

TABLE III-1/X.212

Frames and fields of the X.25/LAPB used to support the OSI CODLS

Frame types	Fields
SABM/SABME	Address field
DISC DM	Address field
I	Information field, N(R), N(S), Address field
RR RNR REJ	N(R)
UA	Address field
FRMR	Information field

Note 1 – All user data fields are octet aligned.

Note 2 – The address field of every type of frame is used to address the appropriate DLSAP.

Note 3 – RR, RNR, REJ and FRMR frames are not directly mapped to the OSI CODLS primitives but are required for the correct operation of the protocol.

III.5.2 *General operation of the X.25/LAPB for supporting the OSI CODLS*

The X.25/LAPB can be used to provide the OSI CODLS in an end system connected to a public or private X.25 PSDN. It can also be used in an environment where the end systems are connected via a dedicated path or via a circuit switched connection.

As shown in Figure III-2/X.212, this DLS provider (more particularly, the DLL entity in the end system) must provide a translation between:

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

b) the frames and associated fields of the X.25/LAPB.

III.6 *Data link connection establishment phase*

III.6.1 *Primitive/parameter and frame/field relationships*

Table III-2/X.212 shows the relationships between the primitives/parameters used during the Data Link Connection establishment Phase and the frames/fields associated with the Data Link Set-Up Procedure.

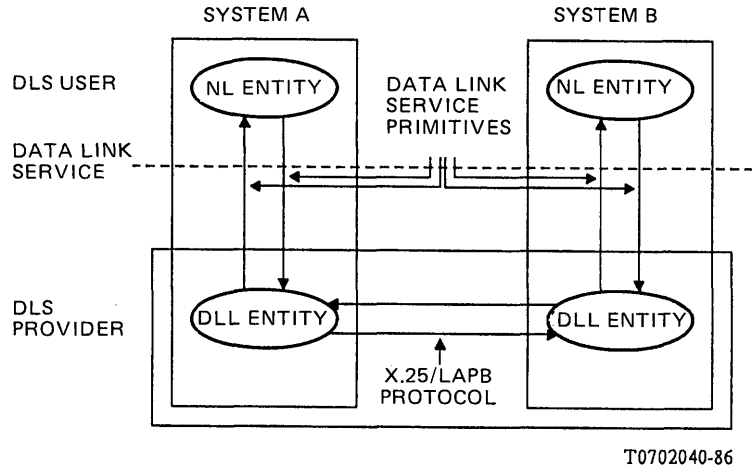


FIGURE III-2/X.212

Relationships between the Data Link Service primitives and LAPB Protocol

TABLE III-2/X.212

CODLS: X.25/LAPB mapping for the Data Link Connection Establishment Phase

CODLS	X.25/LAPB
PRIMITIVES:	FRAMES:
DL-CONNECT request	SABM/SABME (see Note 2)
DL-CONNECT indication	SABM/SABME (see Note 2)
DL-CONNECT response	UA
DL-CONNECT confirm	UA
PARAMETERS:	FIELDS:
Called address	Address field
Calling address	Address field
Responding address	Address field
QOS-parameter set	None (see Note 1)

Note 1 – Since only one level of QOS is available, QOS-parameter set selection is not available when using X.25/LAPB to provide the OSI CODLS.

Note 2 – The frame used is dependent of the type(s) of procedure(s) supported by the DLS provider. The relationships between the throughput QOS-parameter and the use of SABM or SABME when both are supported by the provider is for further study.

III.6.2 *Procedures*

III.6.2.1 *Primitives/frames mapping*

When a DLL entity receives a DL-CONNECT request or DL-CONNECT response primitive from a DLS user, it transmits an SABM/SABME or UA frame, respectively, across the DTE/DXE interface if it can do so before receiving a DL-DISCONNECT request. In the case of a DL-CONNECT response the procedures of III.7.2.1.1 apply.

When a DLL entity in the disconnected phase receives an SABM/SABME frame, it signals a DL-CONNECT indication to the DLS user if it can do so before receiving a DISC frame. When a DLL entity receives in the disconnected phase a UA frame in response to an SABM/SABME frame, it signals a DL-CONNECT confirm to the DLS user unless it has already received a DL-DISCONNECT request from the DLS user.

III.6.2.2 *DLSAP addresses*

There is a maximum of one DLC within a DLSAP at any time. There is a one-to-one relationship between the DLC and the Physical Connection End Point. DLSAP addresses (Called, Calling and Responding) may only take one of the two values A and B as specified in X.25 and X.75.

The Called Address of a DL-CONNECT request primitive is mapped to the Address Field of the corresponding SABM/SABME frame. The Calling Address is not conveyed within the protocol.

The Address Field of a received SABM/SABME frame is mapped to the Called Address of a DL-CONNECT indication primitive. The Calling Address, not conveyed within the protocol, is implicitly deduced or may be implied.

The Responding Address of a DL-CONNECT response primitive is mapped to the Address Field of the corresponding UA frame. The Responding Address is identical to the Called Address in such a point-to-point operation. The Calling Address is not conveyed within the protocol.

The Address Field of a received UA frame is mapped to the Responding Address of a DL-CONNECT confirm primitive. The Calling Address, not conveyed within the protocol, is implicitly deduced or may be implied.

III.6.2.3 *QOS parameter set*

Each X.25/LAPB DLL provides only one value for each QOS subparameter. The DLS user is supposed to have an *a priori* knowledge of the QOS supported by each underlying DLL entity (the QOS may be related to the environment: dedicated path, PSPDN, or circuit switched connection). Consequently, the choice of a DLL entity by a DLS user may be done on the basis of this *a priori* knowledge of the supported QOS.

III.7 *Data link connection release phase*

III.7.1 *Primitive/parameter and frame/field relationships*

Table III-3/X.212 shows the relationships between the primitives/parameters used during the Data Link Connection Release Phase and the frames/fields associated with the Data Link Clearing Procedures.

III.7.2 *Procedures*

III.7.2.1 *Primitive/frame mapping*

III.7.2.1.1 *Connection rejection*

When a DLL entity receives a DL-DISCONNECT request primitive as an answer to the issuance of a DL-CONNECT indication primitive or before transmitting the UA frame as a result of receiving a DL-CONNECT response, it transmits a DM frame across the DTE/DXE interface.

When a DLL entity receives a DM frame in response to the transmission of an SABM/SABME frame, it issues a DL-DISCONNECT indication primitive unless it has already received a DL-DISCONNECT request from the DLS user.

TABLE III-3/X.212

CODLS: X.25/LAPB mapping for the Data Link Connection Release Phase

CODLS	X.25/LAPB
PRIMITIVES: DL-DISCONNECT request DL-DISCONNECT indication	FRAMES: DISC/DM DISC/DM (see Note 2)
PARAMETERS: Originator and reason	FIELDS: None (see Note 1)

Note 1 – Originator is always local when using X.25/LAPB to support the OSI CODLS. Consequently, there is no need to convey a specific parameter within the protocol during the release phase.

Note 2 – DM frame is mapped from a DL-DISCONNECT request primitive if in the connection establishment phase.

III.7.2.1.2 *Connection release from the data transfer phase*

When a DLL entity receives a DL-DISCONNECT request primitive from a DLS user, after it has transmitted a UA frame in response to an SABM/SABME frame, it transmits a DISC frame across the DTE/DXE interface, except if it has previously transmitted a DISC frame because of a protocol error (see below).

If a DL entity detects an error in the operation of the X.25/LAPB for which its action is to clear the link, it transmits a DISC frame across the DTE/DXE interface. If the link is associated with Data Link Connection, it also signals a DL-DISCONNECT indication primitive to the DLS user.

When a DLL entity receives a DISC frame it signals a DL-DISCONNECT indication to the DLS user.

Note – When a DLL entity receives, from the DLS user, a DL-CONNECT request primitive following a DL-DISCONNECT request primitive, the issuing of the SABM/SABME frame is postponed until the DISC frame, related to this DL-DISCONNECT request primitive, has been acknowledged or repeated N2 times.

III.7.2.2 *Originator/Reason*

Originator and Reason parameters have only local significance when using a X.25/LAPB to support the OSI CODLS. They are not conveyed by specific parameters within the protocol during the release phase.

Upon the reception of a DISC or a DM frame:

a) the value of the Originator parameter in the DL-DISCONNECT indication primitive is "unknown"; and

b) the value of the Reason parameter in the DL-DISCONNECT indication primitive is always "reason unspecified".

If the DISC or DM frame is locally issued by the provider:

i) the value of the Originator parameter in the DL-DISCONNECT indication primitive is "provider"; and

ii) the value of the Reason parameter in the DL-DISCONNECT indication primitive is one of those contained in § 13.2.2.a) of this Recommendation.

Consequently, the meaning of the Originator and Reason parameters in a DL-DISCONNECT request primitive is not guaranteed to be the same as in the related DL-DISCONNECT indication primitive.

III.8 *Data transfer phase – Data transfer service*

III.8.1 *Primitive/parameter and frame/field relationships*

Table III-4/X.212 shows the relationships between the primitives/parameters used for the Data Transfer Service and the frames/fields associated with the Data Transfer Procedures.

TABLE III-4/X.212

CODLS: X.25/LAPB mapping for the Data Transfer Service

CODLS	X.25/LAPB
PRIMITIVES: DL-DATA request DL-DATA indication	FRAMES: I I
PARAMETERS: DLS user data	FIELDS: Information field

III.8.2 *Procedures*

III.8.2.1 *Primitives/frames mapping*

When a DLL entity receives a DL-DATA request primitive from a DLS user, it transmits a new I frame across the DTE/DXE interface.

When a DLL entity receives an I frame with N(S) – V(R), it signals a DL-DATA indication primitive to the DLS user.

III.8.2.2 *DLS User Data*

The Data Field of the I frame is directly mapped to the User Data parameters of the DL-DATA request and DL-DATA indication primitives.

III.9 *Data transfer phase – reset service*

III.9.1 *Primitive/parameter and frame/field relationships*

Table III-5/X.212 shows the relationships between the primitives/parameters used for the reset service and the frames/fields associated with the Reset Procedures.

III.9.2 *Procedures*

III.9.2.1 *Primitive/frames mapping*

When a DLL entity receives a DL-RESET request primitive from a DLS user, it transmits an SABM/SABME frame across the DTE/DXE interface.

When a DLL entity receives in the user initiated reset phase a UA frame in response to an SABM/SABME frame, it signals a DL-RESET confirm to the DLS user if it has not already done so (see note to Table III-5/X.212).

TABLE III-5/X.212

CODLS: X.25/LAPB mapping for the Data Link Reset Service

CODLS	X.25/LAPB
PRIMITIVES:	FRAMES:
DL-RESET request	SABM/SABME
DL-RESET indication	SABM/SABME
DL-RESET response	UA
DL-RESET confirm	UA (see Note 1)
PARAMETERS:	FIELDS:
Originator and reason	

Note 1 – Since there is no requirement for a fixed time relationship between the response and confirm primitives, the DLS provider can issue the confirm primitive before receiving the UA frame.

If a DLL entity detects an error in the operation of the X.25/LAPB for which its action is to re-establish the link, it transmits an SABM/SABME frame across the DTE/DXE interface. If the link is associated with a data link connection, it also signals a DL-RESET indication primitive to the DLS user.

When a DLL entity receives, in the local provider initiated reset phase, a UA frame in response to an SABM/SABME frame, no primitive is issued to the DLS user.

When a DLL entity receives an SABM/SABME frame during the data transfer phase, it issues a DL-RESET indication primitive.

When a DLL entity receives a DL-RESET response from a DLS user it transmits a UA frame across the DTE/DXE interface, unless the reset was locally generated.

III.9.2.2 *Originator/Reason*

Originator and Reason parameters have only local significance when using X.25/LAPB to support the OSI CODLS. They are not conveyed by specific parameters within the protocol.

Upon the receiption of a DISC frame:

a) the value of the Originator parameter in the DL-RESET indication primitive is "unknown";

b) the value of the Reason parameter in the DL-RESET indication primitive is always "reason unspecified".

If the DISC frame is locally issued by the provider:

i) the value of the Originator parameter in the DL-RESET indication primitive is "provider"; and

ii) the value of the Reason parameter in the DL-RESET indication primitive is data link flow control congestion, or data link error as appropriate.

Consequently, the meaning of Originator and Reason parameters in a DL-RESET request primitive is not guaranteed to be the same as in the related DL-RESET indication primitive.

NETWORK SERVICE DEFINITION FOR OPEN SYSTEMS INTERCONNECTION FOR CCITT APPLICATIONS[1)]

(Málaga-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.224 specifies the Transport Protocol for Open Systems Interconnection for CCITT applications;

(c) that Recommendation X.210 specifies the OSI Layer Service Definition Conventions for describing the services of the layers of the OSI Reference Model,

unanimously declares

(1) that the scope, field of application, and related definitions and abbreviations of the Open Systems Interconnection Network Service Definition are given in §§ 1 to 4;

(2) that the conventions for describing the Network Service are given in § 5;

(3) that the overview, general characteristics and features of the Network Service, and the classes of Network Service are described in §§ 6, 7 and 8;

(4) that the model of the Network Service is described in § 9;

(5) that the quality of the Network Service is described in § 10;

(6) that the Network Service primitives and their related parameters are defined in §§ 11, 12, 13 and 14;

CONTENTS

[1)] Recommendation X.213 and ISO 8348 [Information Processing Systems – Data Communications – Network Service Definition, Add.2 (Network Layer Addressing), Add.3(Additional Features of the Network Service)] were developed in close collaboration and are technically aligned except for the differences noted in Appendix II.

Introduction

This Recommendation is one of a set of Recommendations produced to facilitate the interconnection of computer systems. It is related to other Recommendations in the set as defined by Recommendation X.200 [1]. The OSI Reference Model subdivides the area of standardization for interconnection into a series of layers of specification, each of a manageable size.

This Recommendation defines the Service provided by the Network Layer to the Transport Layer at the boundary between the Network and Transport Layers of the Reference Model. It provides for the designers of Transport Protocols a definition of the Network Service existing to support the Transport Protocol and for the designers of Network protocols a definition of the services to be made available through the action of the Network Protocol over the underlying service. This relationship is illustrated in Figure 1/X.213.

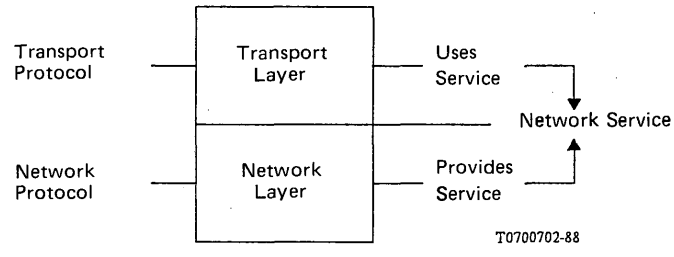


FIGURE 1/X.213

Relationship of the Network Service in this Recommendation to the protocols specified in other OSI Recommendations

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" (Recommendation X.200 [1]). Subnetworks may be either public networks or privately supplied networks. In the case of public networks, their properties may be determined by separate CCITT Recommendations such as Recommendation X.21 for a circuit-switched network or Recommendation X.25 for a packet-switched network.

Throughout the set of OSI Recommendations 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 defined in this Recommendation 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 Recommendations from its use elsewhere to describe the provision of a service by an organization (such as the provision of a service, as defined in other CCITT Recommendations, by an Administration).

Any particular subnetwork may or may not support the OSI Network Service. The OSI Network Service may be provided by a combination of one or more subnetworks and optional additional functions between or outside these subnetworks.

1 Scope and field of application

This Recommendation defines the OSI Network Service 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;

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

The principal objectives of this Recommendation are:

1) to specify the characteristics of a conceptual Network Service and thus, supplement the Reference Model in guiding the development of Network Layer protocols;

2) to encourage convergence of the capabilities offered by providers of subnetworks;

3) to provide a basis for the individual enhancement of existing heterogeneous subnetworks to a common subnetwork-independent Network Service to enable them to be concatenated for the purpose of providing global communication. (Such concatenation may involve optional additional functions which are not defined in this Recommendation.) A definition of the quality of service is an important element of this Recommendation;

4) to provide a basis for the development and implementation of subnetwork-independent Transport Layer protocols decoupled from the variability of underlying public and private subnetworks and their specific interface requirements.

This Recommendation does not specify individual implementations or products nor does it constrain the implementation of entities and interfaces within a system.

There is no conformance of equipment to this Recommendation. Instead, conformance is achieved through implementation of conforming OSI Network protocols which fulfill the Network Service defined in this Recommendation.

2 References

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

Note – See also ISO 7498, Information processing systems – OSI – Basic Reference Model.

[2] Recommendation X.210, OSI Layer Service definition conventions.

Note – See also ISO TR 8509, Information processing systems – Open Systems Interconnection – Service conventions.

[3] Recommendation X.224, Transport Protocol specification for Open Systems Interconnection for CCITT applications.

Note – See also ISO 8073, Information processing systems – OSI – Connection-oriented Transport Protocol specification.

[4] ISO 8348, Information processing systems – Data Communications – Network Service definition.

3 Definitions

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

3.1 *Reference model definitions*

This Recommendation is based on the concepts developed in Recommendation X.200 [1], and makes use of the following terms defined in that Recommendation:

a) expedited Network-Service-data-unit;

b) Network Connection;

c) Network Layer;

d) Network Service;

e) Network-Service-access-point;

f) Network-Service-access-point-address;

g) Network-Service-data-unit;

h) subnetwork.

3.2 *Service conventions definitions*

This Recommendation also makes use of the following terms defined by Recommendation X.210 [2], as they apply to the Network Layer:

a) Network Service user;
b) Network Service provider;
c) primitive;
d) request;
e) indication;
f) response;
g) confirm.

3.3 *Network Service definitions*

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

3.3.1 **calling NS user**

An NS user that initiates an NC establishment request.

3.3.2 **called NS user**

An NS user with whom a calling NS user wishes to establish an NC.

Note – Calling NS users and called NS users defined with respect to a single NC. An NS user can be both a calling and a called NS user simultaneously.

3.3.3 **generic address**

An address which identifies a set of NSAPs rather than a single specific NSAP.

4 Abbreviations

COR	confirmation of receipt
ENSDU	Expedited Network-Service-data-unit
N	Network
NC	Network connection
NL	Network Layer
NS	Network Service
NSAP	Network-Service-access-point
NSDU	Network-Service-data-unit
OSI	Open Systems Interconnection
QOS	quality of service

5 Conventions

5.1 *General conventions*

This Recommendation uses the descriptive conventions given by Recommendation X.210 [2].

The layer service model, service primitives, and time-sequence diagrams taken from those conventions are entirely abstract descriptions; they do not represent a specification for implementation.

5.2 *Parameters*

Service primitives, used to represent service-user/service-provider interactions (see Recommendation X.210 [2]), convey parameters which indicate information available in the user/provider interaction.

The parameters which apply to each group of Network Service primitives are set out in tables in §§ 12 to 14. Each "X" in the tables indicates that the primitive labelling the column in which it falls may carry the parameter labelling the row in which it falls.

Some entries are further qualified by items in brackets. These may be:

a) an indication that the parameter is conditional in some way:

(C) indicates that the parameter is not present on the primitive for every NC; the parameter definition describes the conditions under which the parameter is present or absent;

b) a parameter specific constraint:

(=) indicates that the value supplied in an indication or confirm primitive is always identical to that supplied in the corresponding request or response primitive occurring at the peer NSAP;

c) an indication that some note applies to the entry:

(Note x) indicates that the referenced note contains additional information pertaining to the parameter and its use.

In any particular interface, not all parameters need be explicitly stated. Some may be implicitly associated with the NSAP at which the primitive is issued.

5.3 *NC endpoint identification convention*

If an NS user needs to distinguish among several NCs at the same NSAP, then a local NC endpoint identification mechanism must be provided. All primitives issued at such an NSAP would be required to use this mechanism to identify NCs. Such an implicit identification is not described as a parameter of the service primitives in this Recommendation.

Note – The implicit NC endpoint identification must not be confused with the address parameters of the N-CONNECT primitives (§ 12.2).

6 Overview and general characteristics

The Network Service provides for the transparent transfer of data (i.e., NS-user-data) between NS users. It makes invisible to these NS users the way in which supporting communications resources are utilized to achieve this transfer.

In particular, the Network Service provides for the following:

a) independence of underlying transmission media – The Network Service relieves NS users from all concerns regarding how various subnetworks are used to provide the Network Service. The Network Service hides from the NS user differences in the transfer of data over heterogeneous subnetworks, other than quality of service;

b) end-to-end transfer – The Network Service provides for transfer of NS-user-data between NS users in end systems. All routing and relaying functions are performed by the NS provider including the case where several similar or dissimilar transmission resources are used in tandem or in parallel;

c) transparency of transferred information – The Network Service provides for the transparent transfer of octet-aligned NS-user-data and/or control information. It does not restrict the content, format or coding of the information, nor does it ever need to interpret its structure or meaning;

d) quality of service selection – The Network Service makes available to NS users a means to request and to agree to the quality of service for the transfer of NS-user-data. Quality of service is specified by means of QOS-parameters representing characteristics such as throughput, transit delay, accuracy, and reliability;

e) NS-user-addressing – The Network Service utilizes a system of addressing (NSAP addressing) which allows NS users to refer unambiguously to one another.

7 Features of the Network Service

The Network Service offers the following features to an NS user:

a) the means to establish an NC with another NS user for the purpose of transferring NS-user-data in the form of NSDUs. More than one NC may exist between the same pair of NS users;

b) the establishment of an agreement between the two NS users and the NS provider for a certain QOS associated with each NC;

c) the means of transferring NSDUs in sequence on an NC. The transfer of NSDUs, which consist of an integer number of octets, is transparent, in that the boundaries of NSDUs and the contents of NSDUs are preserved unchanged by the Network Service, and there are no constraints on the NSDU content imposed by the Network Service;

d) the means by which the receiving NS user may flow control the rate at which the sending NS user may send NSDUs;

e) in some circumstances, the means of transferring separate expedited NSDUs in sequence (see § 8). Expedited NSDUs are limited in length and their transmission is subject to a different flow control from normal data across the NSAP;

f) the means by which the NC can be returned to a defined state and the activities of the two NS users synchronized by use of a reset service;

g) in some circumstances, the means for the NS user to confirm the receipt of an NSDU (see § 8);

h) the unconditional, and therefore possibly destructive, release of an NC by either or the NS users or by the NS provider.

8 Classes of Network Service

No distinct classes of Network Service are defined. However, two Network Layer services, Receipt Confirmation and Expedited Data Transfer, are NS provider-options.

A service which is an NS provider-option is one which an NS provider can choose either to provide or not to provide for a particular NC. In circumstances where the NS provider chooses not to provide a provider-option service, it will not be available in the Network Service. If the provider-option Receipt Confirmation or Expedited Data Transfer is provided, it shall be provided as defined in §§ 14.1 to 14.3.

All other Network services are mandatory in the Network Service. Mandatory services shall be provided by every NS provider, and are therefore always available.

9 Model of the network service

9.1 *Model of the Network Layer Service*

This Recommendation uses the abstract model for a layer service defined in § 4 of Recommendation X.210 [2]. The model defines the interactions between the NS users and the NS provider which take place at the two NSAPs. Information is passed between the NS user and the NS provider by service primitives, which may convey parameters.

There are two types of OSI Network Service:

a) a connection-mode Service (defined in §§ 11 to 14 of this Recommendation). The connection-mode Service is characterized by the features a) to h) given in § 7 above;

b) a connectionless-mode Service (for further study).

When making reference to the Network Service, an NS user or NS provider shall state which types of Network Service it expects to use or provide.

9.2 *Model of a Network Connection*

Between the two endpoints of an NC, there exists a flow control function which relates the behaviour of the NS user at one end receiving NS-user-data to the ability of the NS user at the other end to send NS-user-data. As a means of specifying this flow control feature and its relationship with other capabilities provided by the Network Service, the queue model of an NC, described in the following sections, is used.

This queue model of an NC is discussed only to aid in the understanding of the end-to-end service features perceived by users of the Network Service. It is not intended to serve as a substitute for a precise, formal description of the Network Service, nor as a complete specification of all allowable sequences of NS primitives, (Allowable primitive sequences are specified in § 11 – also, see Note below.) In addition, this model does not attempt to describe all the functions or operations of Network Layer entities (including relay entities) which are used to provide the Network Service. No attempt to specify or constrain Network Service implementations is implied.

In interpreting this Recommendation, statements in §§ 12 to 14 concerning the properties of individual primitives have precedence over the general statements in this section.

Note – In addition to the interaction between service primitives described by this model, there may be constraints applied locally on the ability to invoke primitives, as well as service procedures defining particular sequencing constraints on some primitives.

9.2.1 *Queue model concepts*

The queue model represents the operation of an NC in the abstract by a pair of queues linking the two NSAPs. There is one queue for each direction of information flow (see Figure 2/X.213).

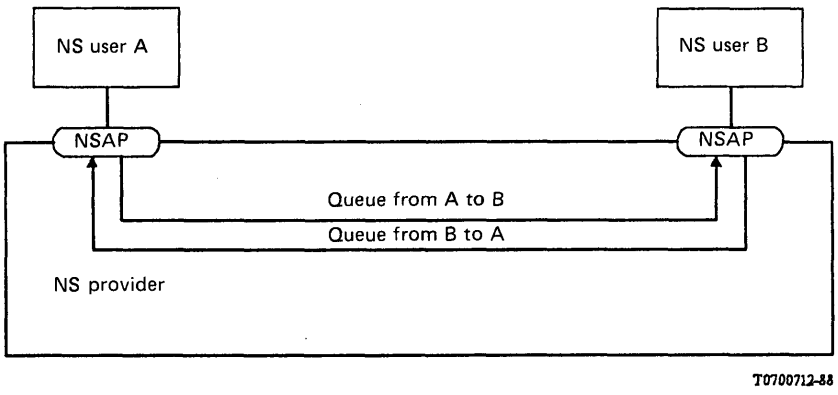


FIGURE 2/X.213

Queue model of a Network Connection

Each queue represents a flow control function in one direction of transfer. The ability of an NS user to add objects to a queue will be determined by the behaviour of the NS user removing objects from that queue and the state of the queue. Objects are entered or removed from the queue, either as the result of interactions at the two NSAPs, or as the result of NS provider initiatives.

The pair of queues is considered to be available for each potential NC.

The objects which may be placed in a queue as a result of interactions at an NSAP (see §§ 12 to 14) are:

a) connect objects (associated with N-CONNECT primitives and all of their parameters);

b) octets of normal NS-user-data (associated with an N-DATA primitive);

c) indications of end-of-NSDU (associated with completion of an N-DATA primitive);

d) expedited NSDUs (associated with N-EXPEDITED-DATA primitives and all their parameters);

e) data acknowledgement objects (associated with N-DATA-ACKNOWLEDGE primitives);

f) reset objects (associated with N-RESET primitives and their parameters);

g) disconnect objects (associated with N-DISCONNECT primitives and all their parameters).

Note – The description of flow control (see § 9.2.3) requires a less abstract description than that used for describing sequences of primitives in §§ 11 to 14. While primitives are defined to be indivisible, for purposes of this queue model, information associated with N-DAT primitives is conceptually subdivided into a sequence of octets "of" NS-user-data followed by an end-of-NSDU indication. This does not imply any particular subdivision in any real interface.

The objects which may be placed in a queue as a result of NS provider initiatives (see §§ 12 to 14) are:

1) reset objects (associated with N-RESET primitives and all their parameters);

2) synchronization mark objects (see § 9.2.4);

3) disconnect objects (associated with N-DISCONNECT primitives and all their parameters).

The queues are defined to have the following general properties:

i) a queue is empty until a connect object has been entered and can be returned to this state, with loss of its contents, by the NS provider (see §§ 9.2.4 and 9.2.5);

ii) objects may be entered into a queue as a result of the actions of the source NS user, subject to control by the NS provider; objects may also be entered into a queue by the NS provider;

iii) objects are removed from the queue under the control of the receiving NS user;

iv) objects are normally removed under the control of the NS user in the same order that they were entered (however see § 9.2.3);

v) a queue has a limited capacity, but this capacity is not necessarily either fixed or determinable.

9.2.2 *NC establishment*

A pair of queues is associated with an NC between two NSAPs when the NS provider receives an N-CONNECT request primitive at one of the NSAPs and a connect object is entered into one of the queues. From the standpoint of one of the NS users of the NC, the queues remain associated with the NC until a disconnect object (associated with an N-DISCONNECT primitive) is either entered or removed from a queue at that NSAP.

If NS user A denotes the NS user who initiates NC establishment (resulting in a connect object being entered into the queue from NS user A to NS user B), then no object other than a disconnect object may be entered into the queue from A to B until after the connect object associated with the N-CONNECT confirm has been removed. In the queue from NS user B to NS user A, objects can be entered only after a connect object associated with an N-CONNECT response from NS user B has been entered; it is possible for a disconnect object to be placed in the queue from B to A instead of a connect object to release the NC.

The properties exhibited by the queues while the NC exists represent the agreements reached among the NS users and the NS provider during the NC establishment procedure concerning quality of service and the use of the receipt and expedited data transfer services.

9.2.3 *Data transfer operations*

Flow control on the NC is represented in this queue model by the management of the queue capacity, allowing objects of certain types to be added to the queues. The conditions affecting entry of reset and disconnect objects are described in item b) below and in §§ 9.2.4 and 9.2.5. The flow control relationship between the other types of objects is summarized by Table 1/X.213.

TABLE 1/X.213

Flow control relationships between queue model objects

The addition of objet x may prevent further addition of object y	Octets of NS-user-data or end-of-NSDU	Expedited NSDU	Data acknowledgement
Octets of normal NS-user-data or end-of-NSDU	Yes	Yes	No
Expedited NSDU	No	Yes	No
Data acknowledgement	No	No	No

Once in the queue, the NS provider may manipulate pairs of adjacent objects, resulting in:

a) change of order – the order of any pair of objects may be reversed, if and only if, the following object is of a type defined to be able to advance ahead of the preceding object. No object is defined to be able to advance ahead of another object of the same type.

b) deletion – any object may be deleted if, and only if, the following object is defined to be destructive with respect to the preceding object. If necessary, the last object in the queue will be deleted to allow a destructive object to be entered. Destructive objects may therefore always be added to the queue. Disconnect objects are defined to be destructive with respect to all other objects. Reset objects are defined to be destructive with respect to all other objects except connect and disconnect objects.

The relationships between objects which may be manipulated as described in a) and b) above are summarized in Table 2/X.213.

Whether the NS provider performs actions resulting in change of order and deletion or not will depend upon the behaviour of the NS users and the agreed QOS for the NC. In general, if an NS user does not cause objects to be removed from a queue, the NS provider shall, after some unspecified period of time, perform all permitted actions of types a) and b).

9.2.4 *Reset operations*

The invocation of a reset procedure is represented in the two queues as follows:

a) Invocation of a reset procedure by the NS provider is represented by the introduction into each queue of a reset object followed by a synchronization mark object.

b) A reset procedure invoked by an NS user is represented by the addition of a reset object to one queue. In this case, the NS provider will insert a reset object followed by a synchronization mark object into the other queue.

The completion of a reset procedure by the issuance of an N-RESET response by an NS user results in a reset object being placed in the queue from the responding NS user.

TABLE 2/X.213

Ordering relationships between queue model objects

Following object x is defined / with respect to preceding object y	Connect	Octets of normal NS user-data	End-of-NSDU	Expedited NSDU	Data acknowledgement	Reset	Synchronization mark	Disconnect
Connect	N/A	– –	N/A	– –	– –	– –	N/A	DES
Octets of normal NS user-data	N/A	– –	– –	AA	AA	DES	N/A	DES
End-of-NSDU	N/A	– –	N/A	AA	AA	DES	N/A	DES
Expedited NSDU	N/A	– –	– –	– –	AA	DES	N/A	DES
Data acknowledgement	N/A	– –	– –	AA	– –	DES	N/A	DES
Reset	N/A	– –	N/A	– –	– –	DES	– –	DES
Synchronization mark	N/A	– –	– –	– –	– –	DES	N/A	DES
Disconnect	N/A	N/A	N/A	N/A	N/A	N/A	N/A	DES

AA indicates that object x is defined to be able in advance ahead of the preceding object y.

DES indicates that object x is defined to be destructive with respect to the preceding object y.

– – indicates that object x is neither destructive with respect to object y nor able to advance ahead of object y.

N/A indicates that object x will not occur in a position succeeding object y in a valid state of a queue.

A synchronization mark object cannot be removed from a queue by an NS user; a queue appears empty to an NS user when a synchronization mark object is the next object in it. Unless destroyed by a disconnect object, a synchronization mark object remains in the queue until the next object following it in the queue is a reset object. Both the synchronization mark object and the following reset object are then deleted by the NS provider.

Note – Associated with the invocation of a reset procedure are restrictions on the issuance of certain other types of primitives. These restrictions will result in restrictions on the entry of certain object types into the queue until the reset procedure is complete.

9.2.5 *NC release*

The insertion into a queue of a disconnect object, which may occur at any time, represents the initiation of an NC release procedure. The release procedure may be destructive with respect to other objects in the two queues and eventually results in the emptying of the queues and the disassociation of the queues with the NC.

The insertion of a disconnect object may also represent the rejection of an NC establishment attempt or the failure to complete NC establishment. In such cases, if a connect object representing an N-CONNECT request primitive is deleted by a disconnect object, then the disconnect object is also deleted. The disconnect object is not deleted when it deletes any other object, including the case where it deletes a connect object representing an N-CONNECT response.

10 Quality of Network Service

The term quality of service (QOS) refers to certain characteristics of an NC as observed between the NC endpoints. QOS describes aspects of an NC which are attributable solely to the NS provider; it can only be properly determined in the absence of NS user behaviour (which is beyond the control of the NS provider) which specifically constrains or impairs the performance of the Network Service.

A value of QOS applies to an antire NC. When determined or measured at both ends of an NC, the QOS observed by the NS users at the two ends of the NC is the same. This is true even in the case of an NC spanning several subnetworks where each subnetwork offers different services.

10.1 *Determination of QOS*

QOS is described in terms of QOS-parameters. The definition of each of these QOS-parameters specifies the way in which the QOS-parameter's value is measured or determined, making reference where appropriate, to primitive events of the NS.

Note 1 – It is important to distinguish the use of the term "QOS-parameters" from the more general term "parameters" as defined in § 5.2 and used throughout this Recommendation. A "QOS-parameter" refers to a specific aspect or component of the QOS for an NC. As described below, a particular QOS-parameter may or may not be related to a parameter defined as part of a Network Service primitive.

Note 2 – For purposes of accuracy and/or convenience, the definition and measurement formula for some QOS-parameters includes a component attributable to the NS user(s). In such cases, the evaluate the QOS attributable solely to the NS provider, this NS user-dependent component must be factored out.

Note 3 – The definition of NS QOS-parameters in terms which provide a means for measurement should not be understood to imply that QOS monitoring or that verification of stated QOS value is, or must be, performed by the NS provider or by the NS users.

It is in terms of the NS QOS-parameters that information about QOS is exchanged among the NS provide and NS users.

Information about the QOS requirements of the NS users may be used by the NS provider for purposes such as protocol selection, route determination, and allocation of resources. Information about the QOS available from the NS provider may be used by NS users for purposes such as selecting QOS enhancement mechanisms and determining the QOS values provided to NS users at higher layers.

The NS QOS-parameters can be divided into two categories as follows:

1) those whose values are "conveyed" between peer NS users by means of the NS during the Establishment phase of an NC. As part of this conveyance, a three-party "negotiation" among the NS users and the NS provider for the purpose of agreeing upon a particular QOS-parameter value may take place; and

2) those whose values are not "conveyed" or "negotiated" among the NS users and the NS provider. For these QOS-parameters, however, information about the values which is useful to the NS provider and each NS user may be made known by local means.

The NS QOS-parameters are defined in §§ 10.2.1 to 10.2.12 below.

The set of NS QOS-parameters that belong to the first category, and the procedures and constraints that apply to conveying and negotiating those QOS-parameters, are specified in § 12.2.7. Once the NS is established, and throughout the lifetime of the NC, the agreed values for these QOS-parameters are not "renegotiated" at any point, and there is no guarantee that the originally negotiated values will be maintained. The NS user should also be aware that, once an NC is established, changes in QOS on the NC are not explicitly signalled in the NS.

For QOS-parameters in the second category, the values for a particular NC are not negotiated, nor are they directly conveyed from NS user to NS user. As a local matter, however, there may be means by which the values of one or more of these QOS-parameters are known and utilized by the NS provider and each NS user. Despite the local nature of particular NS user/NS provider interactions which may occur for the purposes of exchanging QOS-parameter information, the characteristics of an NC which the QOS-parameters describe are applicable and can be observed on a complete NC, end-to-end basis. Thus, in order to give a full characterization of the properties of NCs, the definitions of the entire set of QOS-parameters which apply to the NS, including those classified in category 2, are included in this Recommendation. Other aspects related to category 2 parameters, such as the circumstances of their availability and use, as well as other QOS issues, such as the relationship to OSI management, and multi-layer QOS relationships, are the subjects of other OSI QOS-related specifications.

Note – For non-negotiated QOS-parameters associated with the Data Transfer phase of an NC, when specified, a value of such a QOS-parameter applies to both directions of transfer on the NC.

10.2 *Definition of QOS-parameters*

QOS-parameters can be classified as:

a) QOS-parameters which express Network Service performance, as shown in Table 3/X.213.

b) QOS-parameters that express other Network Service characteristics, as shown in Table 4/X.213.

Note – Some QOS-parameters are defined in terms of the issuance of Network Service primitives. Reference to a primitive in §§ 10.2.1 through 10.2.12 refers to the complete execution of that service primitive at the appropriate NSAP.

10.2.1 *NC establishment delay*

NC establishment delay is the maximum acceptable delay between an N-CONNECT request and the corresponding N-CONNECT confirm primitive.

Note – This delay includes a component, attributable to the called NS user, which is the time between the N-CONNECT indication primitive and the N-CONNECT response.

10.2.2 *NC establishment failure probability*

NC establishment failure probability is the ratio of total NC establishment failures to total NC establishment attempts in a measurement sample.

TABLE 3/X.213

Classification of performance QOS-parameters

Phase	Performance criterion	
	Speed	Accuracy/reliability
NC establishment	NC establishment delay	NC establishment failure probability (misconnection/NC refusal)
Data transfer	Throughput	Residual error rate (corruption, duplication/loss) NC resilience
	Transit delay	Transfer failure probability
NC release	NC release delay	NC release failure probability

TABLE 4/X.213

QOS-parameters not associated with performance

NC protection
NC priority
Maximum acceptable cost

NC establishment failure is defined to occur when a requested NC is not established within the specified maximum acceptable time period as a result of NS provider behaviour such as misconnection, NC refusal, or excessive delay. NC establishment attempts which fail as a result of NS user behaviour such as error, NC refusal, or excessive delay are excluded in calculating NC establishment failure probability.

10.2.3 *Throughput*

Throughput is defined, for each direction of transfer, in terms of a sequence of at least two successfully transferred NSDUs presented continuously to the NS provider at the maximum rate the NS provider can continuously sustain, and unconstrained by flow control applied by the receiving NS user.

Given such a sequence of n NSDUs, where n is greater than or equal to 2, the throughput is defined to be the smaller of:

a) the number of NS-user-data octets contained in the last $n - 1$ NSDUs divided by the time between the first and last N-DATA requests in the sequence; and

b) the number of NS-user-data octets contained in the last $n - 1$ NSDUs divided by the time between the first and last N-DATA indications in the sequence.

Successful transfer of the octets in a transmitted NSDU is defined to occur when the octets are delivered to the intended receiving NS user without error, in the proper sequence, prior to release of the NC by the receiving NS user.

Throughput is specified separately for each direction of transfer. Each throughput specification will specify both the desired "target" value and the minimum acceptable value (i.e., the "lowest quality acceptable") for the NC. (See also § 12.2.7.)

10.2.4 *Transit delay*

Transit delay is the elapsed time between an N-DAT request and the corresponding N-DATA indication. Elapsed time values are calculated only on NSDUs that are successfully transferred.

Successful transfer of an NSDU is defined to occur when the NSDU is transferred from the sending NS user to the intended receiving NS user without error, in the proper sequence, prior to release of the NC by the receiving NS user.

Specification of transit delay will define a pair of values: the desired "target" value and the maximum acceptable (i.e., the "lowest quality acceptable") value. (See also § 12.2.7.) The specified values will be averages and will be based on an NSDU size of 128 octets.

The pair of transit delay values specified for an NC applies to both directions of transfer. That is, the transit delay in each direction is expected to be no worse than that specified.

The transit delay for an individual NSDU may be increased if the receiving NS user exercises flow control. Such occurrences are excluded in calculating both average and maximum Transit Delay values.

10.2.5 *Residual error rate*

Residual error rate is the ratio of total incorrect, lost, and duplicate NSDUs to total NSDUs transferred across the NS boundary during a measurement period. The relationship among these quantities is defined, for a particular NS user pair, as shown in Figure 3/X.213.

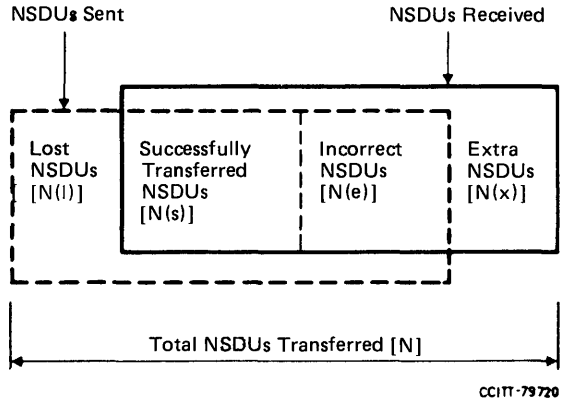
10.2.6 *Tranfer failure probabilty*

Transfer failure probability is the ratio of total transfer failures to total transfer samples observed during a performance measurement.

A transfer sample is a discrete observation of NS provider performance in transferring NSDUs between a specified sending and receiving NS user. A transfer sample begins on input of a selected NSDU at the sending NS user boundary, and continues until the outcome of a given number of NSDU transfer requests has been determined. A transfer sample will normally correspond to the duration of an individual NC.

A transfer failure is a transfer sample in which the observed performance is worse than a specified minimum acceptable level. Transfer failures are identified by comparing the measured values for the supported performance parameters with specified transfer failure thresholds. The three supported performance parameters are throughput, transit delay, and residual error rate.

In systems where Network Service QOS is reliably monitored by the NS provider, transfer failure probability can be estimated by the probability of an NS provider invoked N-DISCONNECT during a transfer sample.



$$RER = \frac{N(e) + N(l) + N(x)}{N}$$

FIGURE 3/X.213

Components of Residual Error Rate

10.2.7 *NC resilience*

NC resilience parameters specify the probability of:

a) an NS provider invoked NC release (i.e., issuance of an N-DISCONNECT indication with no prior N-DISCONNECT request); and

b) an NS provider invoked reset (i.e., issuance of an N-RESET indication with no prior N-RESET request);

during a specified time interval on an established NC.

10.2.8 *NC release delay*

NC release delay is the maximum acceptable delay between an NS user invoked N-DISCONNECT request and the successful release of the NC at the peer NS user. NC release delay is normally specified independently for each NS user. NC release delay does not apply in cases where NC release is invoked by the NS provider.

Issuance of an N-DISCONNECT request by either NS user starts the counting of NC release delay for the other NS user. Successful NC release is signalled to the NS user not initiating the N-DISCONNECT request by an N-DISCONNECT indication.

10.2.9 *NC release failure probability*

NC release failure probability is the ratio of total NC release requests resulting in release failure to total NC release requests included in a measurement sample. NC release failure probability is normally specified independently for each NS user.

A release failure is defined to occur, for a particular NS user, if that user does not receive an N-DISCONNECT indication within the specified maximum NC release delay of the NS user issuing the N-DISCONNECT request (given that the former NS user has not issued an N-DISCONNECT request).

10.2.10 *NC protection*

NC protection is the extent to which an NS provider attempts to prevent unauthorized masquerading or monitoring or manipulation of NS-user-data. NC protection for an NC is specified by selecting any combination of the following features:

a) confidentiality of an entire NSDU sequence on the NC;

b) detection of modification, deletion, replay, or insertion of data within the NSDU sequence on an NC;

c) peer entity authentication. The NS user may request that the NS provider should confirm the identity of the remote NSAP such that there is protection against masquerading by T-entities;

d) authentication of the origin of an NSDU such that there is protection against the unauthorized insertion or replay of the NSDU.

10.2.11 *NC priority*

NC prioriry specifies independently the relative importance of an NC with respect to the following:

a) priority to gain an NC;

b) priority to keep an NC;

c) priority of data on the NC.

NC priority QOS-parameters a) and b) together define the order in which NCs are to be broken to recover resources if necessary. The NS provider is required to accept new requests for NCs with a high prioriry type a) if it can, even if NCs with a lower priority type b) have to be released to do so.

NC priority QOS-parameter c) defines the order in which NCs are to have their QOS degraded. The NCs with a high priority type c) are to have their requests serviced within the required QOS first and remaining resources are then used to attempt to satisfy requests on lower priority NCs.

Note – The use or abuse of the NC priority QOS-parameters can be controlled by one or more of the following:

– user discipline within a closed group of NS users;

– differential tariffs;

– management facilities within the Network Layer such that requests for NC priority are policed and regulated.

10.2.12 *Maximum acceptable cost*

The maximum acceptable cost QOS-parameter specifies the maximum acceptable cost for an NC. The cost may be specified in absolute or relative costs units. The cost of an NC is composed of communications and end-system resource costs.

Note – The possible actions of the NS provider in the event that the maximum acceptable cost for an NC is exceeded are not specified in this Recommendation.

11 Sequence of primitives

This section defines the constraints on the sequences in which the primitives defined in §§ 12 to 14 may occur. The constraints determine the order in which primitives occur, but do not fully specify when they may occur. Other constraints, such as flow control of data, will affect the ability of an NS user or an NS provider to issue a primitive at any particular time.

Table 5/X.213 is a summary of the NS primitives and their parameters.

TABLE 5/X.213

Summary of Network Service primitives and parameters

Phase	Service	Primitive	Parameters
NC establishment	NC establishment	N-CONNECT request	(Called address, calling address, receipt confirmation selection, expedited data selection, QOS-parameter set, NS-user-data)
		N-CONNECT indication	(Called address, calling address, receipt confirmation selection, expedited data selection, QOS-parameter set, NS-user-data)
		N-CONNECT response	(Responding address, receipt confirmation selection, expedited data selection, QOS-parameter set, NS-user-data)
		N-CONNECT confirm	(Responding address, receipt confirmation selection, expedited data selection, QOS-parameter set, NS-user-data)
Data transfer	Data transfer	N-DATA request	(NS-user-data, confirmation request)
		N-DATA indication	(NS-user-data, confirmation request)
	Receipt confirmation (see Note)	N-DATA-ACKNOWLEDGE request	–
		N-DATA-ACKNOWLEDGE indication	–
	Expedited data transfer (see Note)	N-EXPEDITED-DATA request	(NS-user-data)
		N-EXPEDITED-DATA indication	(NS-user-data)
	Reset	N-RESET request	(Reason)
		N-RESET indication	(Originator, reason)
		N-RESET response	–
		N-RESET confirm	–
NC release	NC release	N-DISCONNECT request	(Reason, NS-user-data, responding address)
		N-DISCONNECT indication	(Originator, reason, NS-user-data, responding address)

Note – An NS provider-option service: it may be provided in every Network Service.

11.1 *Relation of primitives at the two NC end points*

A primitive issued at one NC end point will, in general, have consequences at the other NC end point. The relations of primitives of each type to primitives at the other NC end point are defined in the appropriate §§ 12 to 14; all these relations are summarized in the diagrams in Figure 4/X.213.

However, an N-DISCONNECT request or indication primitive may terminate any of the other sequences before completion. An N-RESET request or indication may terminate a data transfer, expedited data transfer, or receipt confirmation sequence before completion.

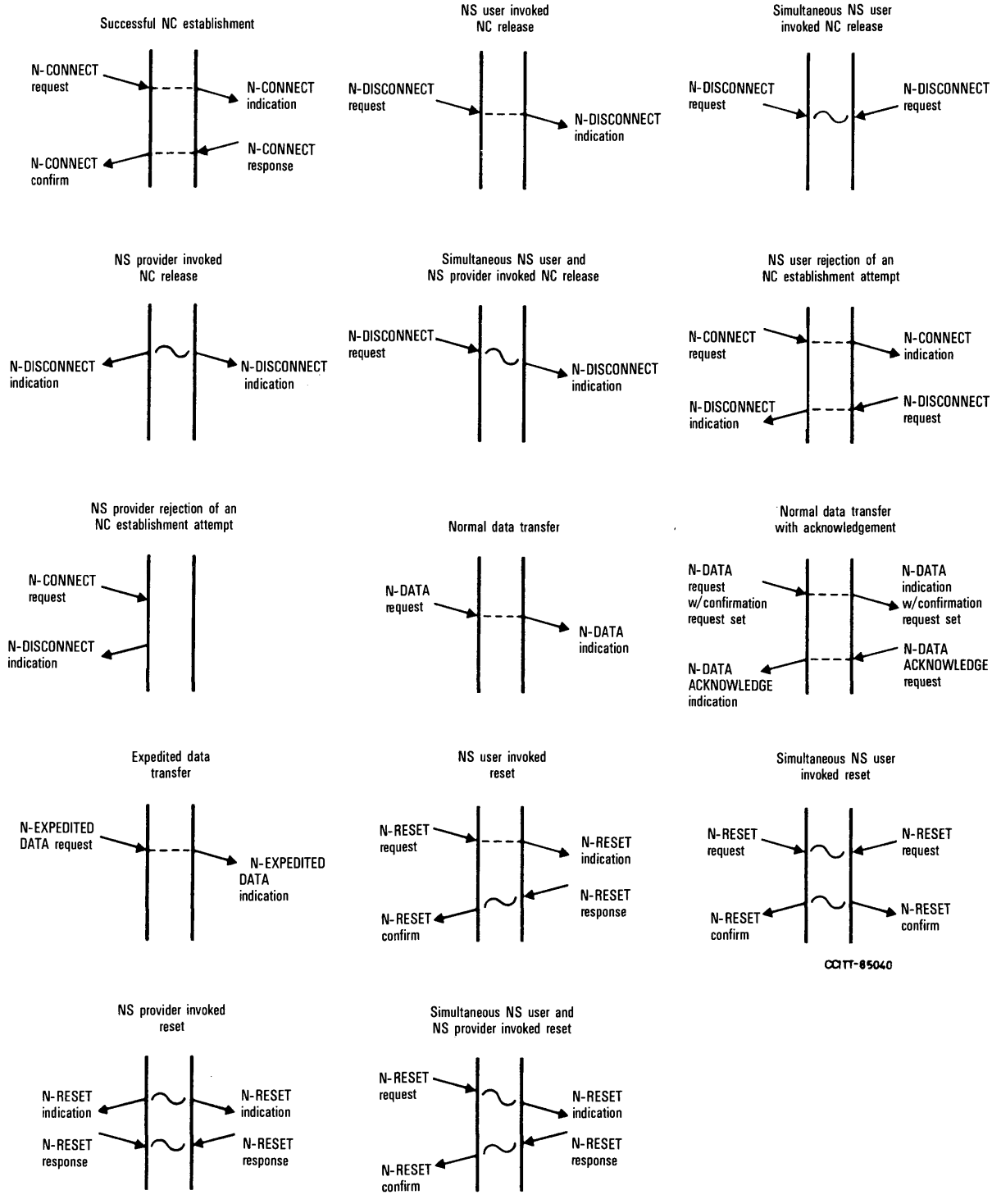


FIGURE 4/X.213

Summary of network service primitive time sequence diagrams

11.2 *Sequence of primitives at one NC endpoint*

The possible overall sequences of primitives at an NC endpoint are defined in the state transition diagram, Figure 5/X.213. In the diagram:

a) a primitive which is not shown as resulting in a transition (from one state to the same state, or from one state to a different state) is not permitted in that state (however, see § 11.1 above concerning the effect of N-DISCONNECT and N-RESET primitives);

b) N-DISCONNECT stands for either the request or the indication form of the primitive in all cases;

c) the labelling of the states NS user invoked reset pending (state 5) and NS provider invoked reset pending (state 6) indicates the party which started the local interaction, and does not necessarily reflect the value of the originator parameter in the associated N-RESET primitive;

d) the Idle state (state 1) reflects the absence of an NC. It is the initial and final state of any sequence, and once it has been re-entered, the NC is released;

e) the use of a state transition diagram to describe the allowable sequences of service primitives does not impose any requirements or constraints on the internal organization of any implementations of the Network Service.

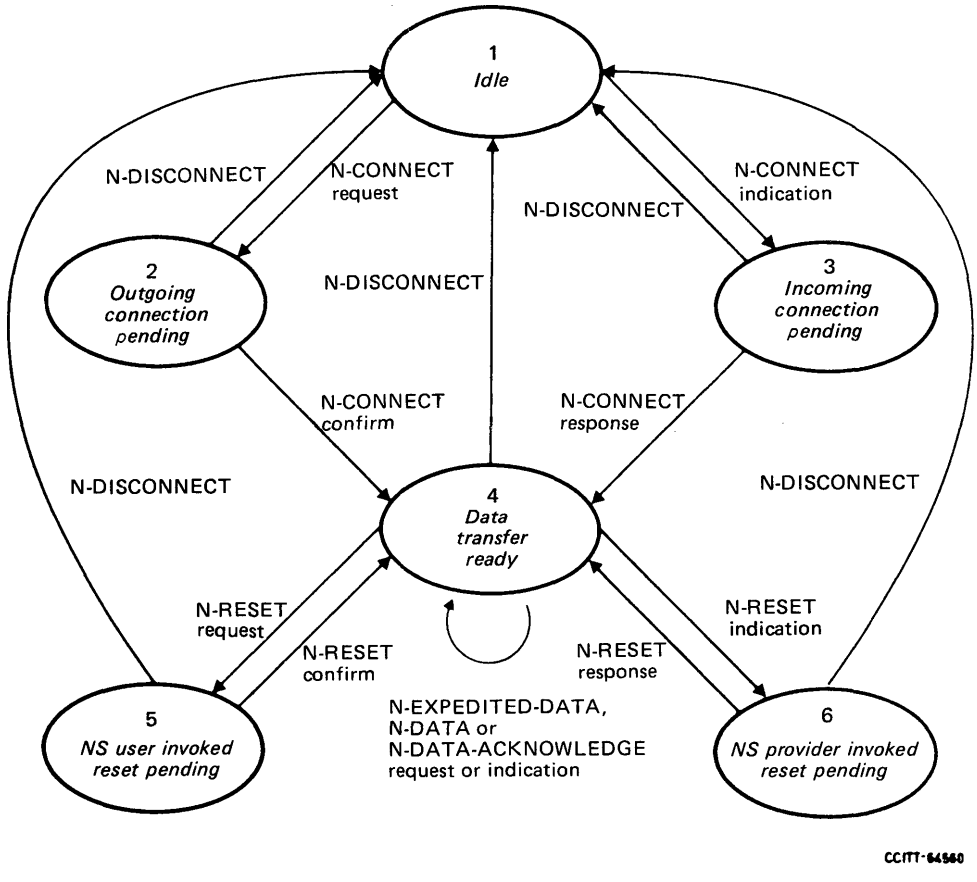


FIGURE 5/X.213

State transition diagram for sequences of primitives at an NC end point

12 Network connection establishment phase

12.1 *Function*

The NC establishment service primitives can be used to establish an NC, provided the NS users exist and are known to the NS provider.

Simultaneous N-CONNECT requests at the two NSAPs are handled independently by the NS provider; they may result in two, one or zero NCs.

12.2 *Types of primitives and parameters*

Table 6/X.213 indicates the types of primitives and the parameters needed for NC establishment.

TABLE 6/X.213

NC establishment primitives and parameters

Parameter \ Primitive	N-CONNECT request	N-CONNECT indication	N-CONNECT response	N-CONNECT response
Called address	X	X(=) (see Note)		
Calling address	X (see Note)	X(=)		
Responding address			X (see Note)	X(=)
Receipt confirmation selection	X	X	X	X(=)
Expedited data selection	X	X	X	X(=)
QOS-parameter set	X	X	X	X(C=)
NS-user-data	X(C)	X(C=)	X(C)	X(C=)

Note – This parameter may be implicitly associated with the NSAP at which the primitive is issued.

12.2.1 *Addresses*

The parameters which take addresses as values (§§ 12.2.2 to 12.2.4) all refer to NSAP addresses. The NSAP address parameters will accommodate variable length addresses up to a defined maximum of 40 decimal digits (when expressed in decimal digit syntax). Network Layer Addressing is specified in Annex A.

The values of these addresses as supplied by the NS user are not necessarily checked or authenticated by the NS provider. An NS user receiving these addresses in N-CONNECT indication or confirm primitives can only rely on their validity if the NS user has knowledge that the NS provider guarantees address correctness.

Note – Mechanisms operating within the NS provider, such as call redirection or resolution of generic addresses, may result in address parameters in corresponding primitives not being identical in the following cases:

a) the responding address parameter on the N-CONNECT response may not necessarily be the same as the called address parameter on the N-CONNECT indication;

b) the responding adress parameter on the N-CONNECT confirm may not necessarily be the same as the called address parameter on the N-CONNECT request.

12.2.2 *Called address parameter*

The called address parameter conveys an address identifying the NSAP to which the NC is to be established. Where explicitly supplied, the addresses in corresponding N-CONNECT resquest and indication primitives are identical.

12.2.3 *Calling address parameter*

The calling address parameter conveys the address of the NSAP from which the NC has been requested. Where explicitly supplied, the addresses in corresponding N-CONNECT request and indication primitives are identical.

12.2.4 *Responding address parameter*

The responding address parameter conveys the address of the NSAP to which the NC has been established. Where explicitly supplied, the addresses in corresponding N-CONNECT response and confirm primitives are identical. This parameter always conveys a specific NSAP address and not a generic NSAP address.

12.2.5 *Receipt confirmation selection parameter*

The receipt confirmation selection parameter indicates the use/availability of the receipt confirmation service on the NC. If the receipt confirmation service is not provided in the Network Service, then it cannot be used on the NC (see § 8). The value of this parameter is either "user of receipt confirmation" or "no use of receipt confirmation". The values on the various primitives are related such that:

a) on the N-CONNECT request, either of the defined values may occur;

b) on the N-CONNECT indication, the value is either equal to the value on the request primitive, or is "no use of receipt confirmation";

c) on the N-CONNECT response, the value is either equal to the value on the indication primitive or is "no use of receipt confirmation";

d) on the N-CONNECT confirm, the value is equal to the value on the response primitive.

Since receipt confirmation may not be provided in the Network Service and since, when it is available, both NS users and the NS provider must agree to its use, there are four possible cases of negotiation of receipt confirmation on an NC:

i) the calling NS user does not request it – it is not used;

ii) the calling NS user requests it but the NS provider does not provide it – it is not used;

iii) the calling NS user requests it and the NS provider agrees to provide it, but the called NS user does not agree to its use – it is not used;

iv) the calling NS user requests it, the NS provider agrees to provide it, and the called NS user agrees to its use – it can be used.

12.2.6 *Expedited data selection parameter*

The expedited data selection parameter indicates the use/availability of the expedited data transfer service on the NC. If the expedited data transfer service is not available from the NS provider (see § 8), then it cannot be used on the NC. The value of this parameter is either "use of expedited data" or "no use of expedited data". The values on the various primitives are related such that:

a) on the N-CONNECT request, either of the defined values may occur;

b) on the N-CONNECT indication, the value is either equal to the value on the request primitive, or is "no use of expedited data";

c) on the N-CONNECT response, the value is either equal to the value on the indication primitive or is "no use of expedited data";

d) on the N-CONNECT confirm, the value is equal to the value on the response primitive.

12.2.7 *QOS-parameter set*

For each QOS-parameter which is conveyed during NC establishment, a set of "subparameters" is defined from among the following possibilities:

i) a "target" value which is the QOS value desired by the calling NS user;

ii) the "lowest quality acceptable" value which is the lowest QOS value agreeable to the calling NS user;

iii) an "available" value which is the QOS value the NS provider is willing to provide; and

iv) a "selected" value which is the QOS value to which the called NS user agrees.

The set of values which can be specified for each subparameter is defined in every Network Service. Each set of values includes the value "unspecified". It may also include a value defined to be a "default" value, which is mutually understood by the NS provider and the NS user between which it is conveyed.

Note – "Default" values are defined between a particular NS user and the NS provider. Different "defaults" may exist for different NS users and thus a value which is understood as a "default" at one end of an NC may not be the "default" value at the other end.

In those cases where both the subparameters "target" and "lowest quality acceptable" are specified by the calling NS user, they are boundary parameters defining a range of QOS values to which the calling NS user will agree. Similarly, where both the subparameters "available" and "lowest quality acceptable" are specified by the NS provider, they are boundary parameters defining a range of QOS values which the NS provider is willing to provide. These ranges are defined to include the values of both of the boundary subparameters, plus any values allowed for these subparameters which lie between the boundary subparameters. In the case where the "target" (or the "available") subparameter has a specified value but the "lowest quality acceptable" value is "unspecified", the rage is defined to consist of the "target" value plus all other values which are allowed for these subparameters and which are lower (in QOS terms) than the "target". If the value for both the "target" and "lowest quality acceptable" is "unspecified", then no range of values is defined.

Note – For other value assignments (e.g. "target" is "unspecified" but "lowest quality acceptable" has a specified value), the range is not defined since these assignments are not allowed in the negotiation procedures described in §§ 12.2.7.1 and 12.2.7.2.

12.2.7.1 *Throughput*

Table 7/X.213 indicates the presence of the QOS-subparameters for the throughput QOS-parameters in the N-CONNECT primitives.

The negotiation and conveyance of each of the two throughput QOS-parameters are conducted as follows:

a) In the N-CONNECT request primitive, the calling NS user specifies values for the "target" and "lowest quality acceptable" (i.e. lowest throughtput) subparameters. Permitted value assignments are:

Case 1: both the "target" and "lowest quality acceptable" are "unspecified";

Case 2: values other than "unspecified" are specified for both "target" and "lowest quality acceptable";

Case 3: a value other than "unspecified" is specified for the "target" and the "lowest quality acceptable" is "unspecified".

Note – The case where "target" is "unspecified" and the "lowest quality acceptable" 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" (case 2).

b) If the value assignment of the "target" and "lowest quality acceptable" subparameters are as defined in case 1, then the NS provider determines the highest QOS throughput value which is to be offered on the NC. This value (which may be the "default" value understood by the NS provider and the called NS user) is specified as the "available" subparameter in the N-CONNECT indication while the "lowest quality acceptable" subparameter value is "unspecified". If the requested QOS value assignements are as defined in case 2 or case 3, then, if the NS provider does not agree to provide a QOS in the requested range, the NC establishment attempt is rejected as described in § 13.5. If the NS provider does agree to provide a QOS in the requested range, then in the N-CONNECT indication, the "available" subparameter specifies the highest QOS value within the range which the NS provider is willing to provide and the "lowest quality acceptable" subparameter value is identical to that of the "lowest quality acceptable" subparameter in the N-CONNECT request.

c) If the called NS user does not agree to a QOS in the range between the "available" and the "lowest quality acceptable" subparameters of the N-CONNECT indication then the NS user rejects the NC establishment attempt as described in § 13.4.

d) If the called NS user does agree to a QOS in the specified range, then the NS user specifies the agreed to value in the "selected" parameter on the N-CONNECT response.

e) In the N-CONNECT confirm, the "selected" subparameter has a value identical to that of "selected" in the N-CONNECT indication.

A summary of the negotiation procedures for the throughput QOS-subparameters is contained in Table 8/X.213.

TABLE 7/X.213

Negotiated QOS-subparameters for throughput QOS-parameters

Parameter \ Primitive	N-CONNECT request	N-CONNECT indication	N-CONNECT response	N-CONNECT confirm
Throughput 1 "target" (calling to called)	X(=)			
Throughput 1 "lowest quality acceptable" (calling to called)	X	X(=)		
Throughput 2 "target" (called to calling)	X			
Throughput 2 "lowest quality acceptable" (called to calling)	X	X(=)		
Throughput 1 "available" (calling to called)		X		
Throughput 2 "available" (called to calling)		X		
Throughput 1 "selected" (calling to called)			X	X(=)
Throughput 2 "selected" (called to calling)			X	X(=)

TABLE 8/X.213

Negotiation of throughput QOS-subparameters

	Calling NS user specifies in N-CONNECT request		NS provider specifies in N-CONNECT indication		Called NS user specifies in N-CONNECT response	NS provider specifies in N-CONNECT confirm	Notes
	"Target"	"Lowest quality acceptable"	"Available"	"Lowest quality acceptable"	"Selected"	"Selected"	
Case 1	"Unspecified"	"Unspecified"	Z	"Unspecified"	A	A	Z may be a "default" value $Z \geqslant A > 0$
Case 2	X	Y	Z	Y	A	A	X and/or Y may be defined to be the "default" value at the calling NS user end, called NS user end, or both $X \geqslant Z \geqslant Y$; $Z \geqslant A \geqslant Y$
Case 3	X	"Unspecified"	Z	"Unspecified"	A	A	X may be a "default" value $X \geqslant Z > 0$; $Z \geqslant A > 0$

12.2.7.2 *Transit delay*

Note – The implementation of the transit delay negotiation requires urgent further study in order to have a harmonized realization in different types of sub-networks. Special attention is required as regards routing and charging consequences.

Table 9/X.213 indicates the presence of the QOS-subparameters for the transit delay QOS-parameter in the N-CONNECT primitives.

The negotiation and conveyance of the transit delay QOS-parameter are conducted as follows:

a) In the N-CONNECT request primitive, the calling NS user specifies values for the "target" and "lowest quality acceptable" (i.e. highest acceptable transit delay) subparameters. Permitted value assignements are:

   Case 1: both the "target" and "lowest quality acceptable" are "unspecified";

   Case 2: values other than "unspecified" are specified for both "target" and "lowest quality acceptable";

   Case 3: a value other than "unspecified" is specified for the "target" and the "lowest quality acceptable" is "unspecified".

   Note – The case where "target" in "unspecified" and the "lowest quality acceptable" 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".

b) If the value assignments of the "target" and "lowest quality acceptable" subparameters are as defined in case 1, then the NS provider determines the transit delay value to be offered on the NC and specifies it as the "available" subparameter in the N-CONNECT indication.

   If the value assignments are as defined in case 2 or case 3, then if the NS provider does not agree to provide a QOS in the requested range, the NC establishment attempt is rejected as described in § 13.5. If the NS provider does agree to provide a QOS in the requested range, the "available" subparameter in the N-CONNECT indication specifies the value of QOS which is offered.

c) If the called NS user does not agree to the QOS specified as "available", the NS user rejects the NC establishment attempt as described in § 13.4.

d) If the called NS user does agree to the "available" QOS, then the NS user issues an N-CONNECT response (the N-CONNECT response does not convey any transit delay QOS-subparameters).

e) In the N-CONNECT confirm the "selected" subparameter value is identical to that specified as "available" in the N-CONNECT indication.

A summary of the negotiation procedures for the transit delay QOS-subparameters is contained in Table 10/X.213.

TABLE 9/X.213

Negotiated QOS-subparameters for transit delay QOS-parameter

Parameter \ Primitive	N-CONNECT request	N-CONNECT indication	N-CONNECT response	N-CONNECT confirm
Transit delay "target"	X			
Transit delay "lowest quality acceptable"	X			
Transit delay "available"		X		
Transit delay "selected"				X

TABLE 10/X.213

Negotiation of transit delay QOS-subparameters

	Calling NS user specifies in N-CONNECT request		NS provider specifies in N-CONNECT indication	Called NS user specifies in N-CONNECT response	NS provider specifies in N-CONNECT confirm	Notes
	"Target"	"Lowest quality acceptable"	"Available"		"Selected"	
Case 1	"Unspecified"	"Unspecified"	Z		Z	
Case 2	X	Y	Z		Z	X and/or Y may be a "default" value $X \leqslant Z \leqslant Y$
Case 3	X	"Unspecified"	Z		Z	X may be a "default" value $X \leqslant Z < \infty$

12.2.7.3 *NC Protection*

The values and meaning of the NC-protection QOS parameter are given in § 10.2.10. Table 11/X.213 indicates the presence of the QOS-subparameter for the NC Protection QOS-parameter in the N-CONNECT primitives.

TABLE 11/X.213

Negotiated QOS-subparameter for NC Protection QOS-parameters

Parameter \ Primitive	N-CONNECT request	N-CONNECT indication	N-CONNECT response	N-CONNECT confirm
NC Protection "Target"	X			
NC Protection "Lowest Quality Acceptable"	X	X(=)		
NC Protection "Available"		X		
NC Protection "Selected"			X	X(=)

The negotiation and conveyance of the NC Protection QOS parameter is conducted as follows:

a) In the N-CONNECT request primitive, the calling NS user specifies values for the "Target" and "Lowest Quality Acceptable" subparameters; permitted value assignments are:

Case 1: both the "Target" and "Lowest Quality Acceptable" are "unspecified"

Case 2: values other than "unspecified" are specified for both "Target" and "Lowest Quality Acceptable"

Case 3: a value other than "unspecified" is specified for the "Target" and the "Lowest Quality Acceptable" is "unspecified"

Note – The case where "Target" is "unspecified" and the "Lowest Quality Acceptable" 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" (case 2).

b) If the NS provider does not support a choice of NC Protection levels then the value of the "Target" subparameter is conveyed by the NS provider and passed to the called NS user unchanged as the "Available" subparameter in the N-Connect indication.

c) If the NS provider does support a choice of NC Protection levels, then

1) In case 1

the NS provider determines the QOS value to be offered on the NC and specifies it in the "Available" subparameter in the N-CONNECT indication.

2) In cases 2 and 3

if the NS provider does not agree to provide a QOS in the requested range, then the NC establishment attempt is rejected as described in § 13.5. If the NS provider does agree to provide a QOS in the requested range, then in the N-CONNECT indication, the "Available" subparameter specifies the highest QOS value within the range which the NS provider is willing to provide.

d) The value of the "Lowest Quality Acceptable" subparameter in the N-CONNECT indication is identical to that in the N-CONNECT request.

e) If the value of the "Available" subparameter of the N-CONNECT indication is "unspecified" then

1) If the called NS user does not agree to accept establishment of a connection with this unspecified quality, the NS user rejects the NC establishment attempt as described in § 13.4.

2) If the called NS user does agree, then the NS user specifies the value "unspecified" in the "Selected" subparameter of the N-CONNECT response.

Note – When connection is established with the value of "unspecified" selected, it follows that the QOS provided may be at any level at the discretion of the Network Service provider. Consequently, the called NS user would agree to such a connection only if any level of QOS, even the lowest, is acceptable.

f) If the value of the "Available" subparameter in the N-CONNECT indication is not "unspecified" then

1) If the called NS user does not agree to a QOS in the range identified by the "Available" and "Lowest Quality Acceptable" subparameters of the N-CONNECT indication then the NS user rejects the NC establishment attempt as described in § 13.4.

2) If the called NS user does agree to a QOS in the identified range, then the NS user specifies the agreed value in the "Selected" subparameter of the N-CONNECT response.

g) In the N-CONNECT confirm, the "Selected" subparameter has a value identical to that of "Selected" in the N-CONNECT response.

12.2.7.4 *NC Priority*

Section 10.2.11 specifies the values and meaning of NC Priority QOS parameters. This section specifies the conveyance of these parameters and applies to each of the three independent aspects of NC Priority defined in § 10.2.11.

Table 12/X.213 indicates the presence of the QOS-subparameters for the NC Priority QOS-parameter in the N-CONNECT primitives.

TABLE 12/X.213

Negotiated QOS-subparameters for NC Priority QOS-parameter

Parameter \ Primitive	NC-CONNECT request	NC-CONNECT indication	NC-CONNECT response	NC-CONNECT confirm
NC Priority "Target"	X			
NC Priority "Lowest Quality Acceptable"	X	X(=)		
NC Priority "Available"		X		
NC Priority "Selected"			X	X(=)

The conveyance of NC Priority QOS-parameter is conducted as follows:

a) In the N-CONNECT request primitive, the calling NS user specifies values for the "Target" and "Lowest Quality Acceptable" subparameters; permitted value assignments are:

Case 1: both the "Target" and "Lowest Quality Acceptable" are "unspecified".

Case 2: values other than "unspecified" are specified for both "Target" and "Lowest Quality Acceptable".

Case 3: a value other than "unspecified" is specified for the "Target" and the "Lowest Quality Acceptable" is "unspecified".

Note – The case where "Target" is "unspecified" and the "Lowest Quality Acceptable" has a value other than "unspecified" is not permitted; logically this case can be represented by the permitted assignement where an identical value is specified for both the "Target" and "Lowest Quality Acceptable" (case 2).

b) If the NS provider does not support a choice of NC Priority levels then the value of the "Target" subparameter is conveyed by the NS provider and passed to the called NS user unchanged as the "Available" subparameter in the N-CONNECT indication.

c) If the NS provider does support a choice of NC Priority levels, then

1) In case 1

the NS provider determines the QOS value to be offered on the NC and specifies it in the "Available" subparameter in the N-CONNECT indication.

2) In cases 2 and 3

if the NS provider does not agree to provide a QOS in the requested range, then the NC establishment attempt is rejected as described in § 13.5. If the NS provider does agree to provide a QOS in the requested range, then in the N-CONNECT indication, the "Available" subparameter specifies the highest QOS value within the range which the NS provider is willing to provide.

d) The value of the "Lowest Quality Acceptable" subparameter in the N-CONNECT indication is identical to that in the N-CONNECT request.

e) If the value of the "Available" subparameter of the N-CONNECT indication is "unspecified" then

1) If the called NS user does not agree to accept establishment of a connection with this unspecified quality, the NS user rejects the NC establishment attempt as described in § 13.4.

2) If the called NS user does agree, then the NS user specifies the value "unspecified" in the "Selected" subparameter of the N-CONNECT response.

Note – When connection is established with the value "unspecified" selected, it follows that the QOS provided may be at any level at the discretion of the Network Service provider. Consequently, the called NS user would agree to such a connection only if any level of QOS, even the lowest, is acceptable.

f) If the value of the "Available" subparameter in the N-CONNECT indication is not "unspecified" then

1) If the called NS user does not agree to a QOS in the range identified by the "Available" and "Lowest Quality Acceptable" subparameters of the N-CONNECT indication then the NS user rejects the NC establishment attempt as described in § 13.4.

2) If the called NS user does agree to a QOS in the identified range, then the NS user specifies the agreed value in the "Selected" subparameter of the N-CONNECT response.

g) In the N-CONNECT confirm, the "Selected" subparamter has a value identical to that of "Selected" in the N-CONNECT response.

12.2.8 *NS-user-data parameter*

The NS-user-data parameter allows the transfer of NS-user-data between NS users, without modification by the NS provider. The NS user may send any integer number of octets of NS-user-data between zero and 128 octets inclusive.

Note – The objective is to make this parameter a mandatory parameter to be supported by all sub-networks in the future. However, a number of existing sub-networks cannot support it now. During the interim period, while these sub-networks exist and are not modified to provide this parameter, it is considered as a provider-option. No negotiation mechanism is needed in the Network Service. Limiting, in some sub-networks, the length of NS-user-data to be provided to a value lower than 128 octets (e.g. 16 to 32 octets) for an interim period would imply fewer changes to existing interfaces and signalling systems and would simplify the introduction of such a service in existing sub-networks.

12.3 *Sequence of primitives*

The sequence of primitives in a successful NC establishment is defined by the time sequence diagram in Figure 6/X.213.

The NC establishment procedure may fail either due to the inability of the NS provider to establish an NC or due to the unwillingness of the called NS user to accept an N-CONNECT indication (for these cases, see NC release service, §§ 13.4 and 13.5). In addition, the NC establishment attempt may be aborted by the NS provider or either of the NS users at any other time before the issuing of the N-CONNECT confirm.

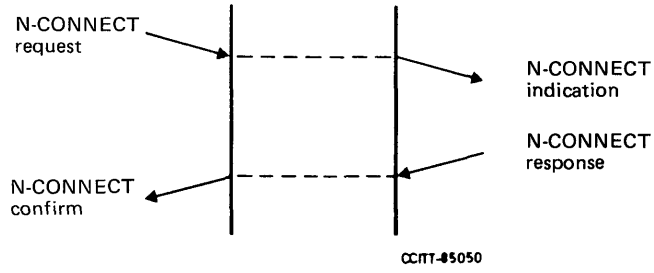


FIGURE 6/X.213

Sequence of primitives in successful NC establishment

13 Network Connection Release Phase

13.1 *Function*

The NC release service primitives are used to release an NC. The NC release may be performed:

a) by either or both of the NS users to release an established NC;

b) by the NS provider to release an established NC. All failures to maintain an NC are indicated in this way;

c) by the called NS user to reject an N-CONNECT indication;

d) by the NS provider to indicate its inability to establish a requested NC.

NC release is permitted at any time regardless of the current phase of the NC. Once an NC release procedure has been invoked, the NC will be released; a request for NC release cannot be rejected. After NC release has been invoked at one NC endpoint, the NS provider may discard any normal or expedited NS-user-data that has not yet been delivered at the other NC endpoint and may cause any uncompleted sequence of primitives for NC establishment, receipt confirmation, or reset to remain uncompleted.

13.2 *Types of primitive and parameters*

Table 13/X.213 indicates the types of primitives and the parameters needed for NC Release.

TABLE 13/X.213

NC release primitives and parameters

Parameter \ Primitive	N-DISCONNECT request	N-DISCONNECT indication
Originator		X
Reason	X	X
NS-user-data	X(C)	X(C=)
Responding address	X(C) (see Note)	X(C=)

Note – This parameter may be implicitly associated with the NSAP at which the primitive is issued.

13.2.1 *Originator parameter*

The originator parameter indicates the source of the NC release. Its value indiates either the "NS user", "NS provider", or "undefined".

Note – The value "undefined" is not permitted when an N-DISCONNECT indication is issued by an NS user or an NS provider in order to reject an NC establishment attempt (see §§ 13.4 and 13.5).

13.2.2 *Reason parameter*

The reason parameter gives information about the cause of the NC release. The value conveyed in this parameter will be as follows:

a) When the originator parameter indicates an NS provider invoked release, the value is one of:
   1) disconnection-permanent condition;
   2) disconnection-transient condition;
   3) connection rejection-NSAP address unknown (permanent condition);

4) connection rejection-NSAP unreachable/transient condition;

5) connection rejection-NSAP unreachable/permanent condition;

6) connection rejection-QOS not available/permanent condition;

7) connection rejection-QOS not available/transient condition;

8) connection rejection-reason unspecified/permanent condition;

9) connection rejection-reason unspecified/transient condition;

b) When the originator parameter indicates an NS user invoked release, the value is one of:

1) disconnection-normal condition;

2) disconnection-abnormal condition;

3) connection rejection-permanent condition;

4) connection rejection-transient condition;

5) connection rejection-QOS not available/transient condition;

6) connection rejection-QOS not available/permanent condition;

7) connection rejection-incompatible information in NS-user-data.

c) When the originator parameter value is "undefined", then the value of the reason parameter shall also be "undefined".

13.2.3 *NS-user-data parameter*

The NS-user-data parameter allows the transfer of NS-user data between NS users, without modification by the NS provider. An NS user invoking NC release may send any integer number of octets of NS-user-data between zero and 128 inclusive. In an N-DISCONNECT indication, this parameter can have a non-zero number of octets of NS-user-data only if the originator parameter has the value of "NS user".

The NS-user-data sent is lost if NC release is simultaneously invoked by either the NS provider or the intended receiving NS user (§ 13.3).

Note – The objective is to make this parameter a mandatory parameter to be supported by all sub-networks in the future. However, a number of existing sub-networks cannot support it now. During the interim period, while these sub-networks exist and are not modified to provide this parameter, it is considered as a provider-option. No negotiation mechanism is needed in the Network Service.

13.2.4 *Responding address parameter*

The responding address parameter is present in this primitive only in the case where the primitive is used to indicate rejection of an NC establishment attempt by an NS user (see § 13.4). The parameter conveys the address of the NSAP from which the N-DISCONNECT request was issued and, where explicitly supplied, the addresses in the corresponding request and indication primitives are identical. Under certain circumstances (e.g. call redirection, generic addressing, etc.) this address may be different from the "called address" in the corresponding N-CONNECT request primitive.

13.3 *Sequence of primitives when releasing an established NC*

The sequence of primitives depends on the origin or origins of the NC release action. The sequence may be:

a) invoked by one NS user, with a request from that NS user leading to an indication to the other NS user;

b) invoked by both NS users, with a request from each of the NS users;

c) invoked by the NS provider, with an indication to each of the NS users;

d) invoked independently by one NS user and the NS provider, with a request from the originating NS user and an indication to the other NS user.

The sequences of primitives in these four cases are expressed in the time sequence diagrams in Figure 7/X.213 to 10/X.213.

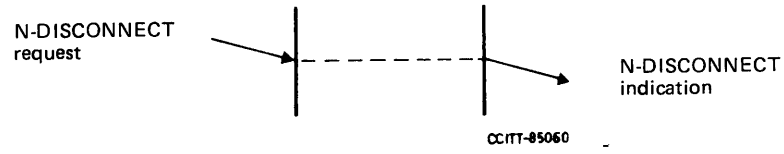


FIGURE 7/X.213

Sequence of primitives in NS user invoked NC release

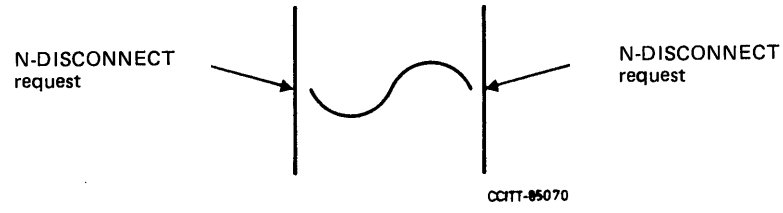


FIGURE 8/X.213

Sequence of primitives in simultaneous NS user invoked NC release

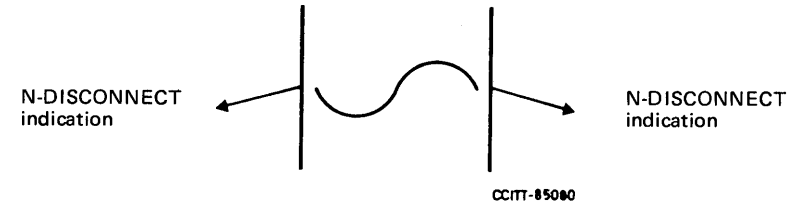


FIGURE 9/X.213

Sequence of primitives in NS provider invoked NC release

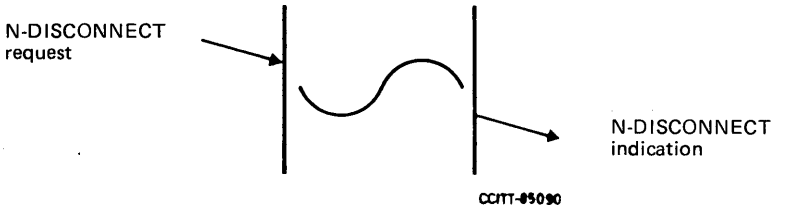


FIGURE 10/X.213

Sequence of primitives in simultaneous NS user and NS provider invoked NC release

13.4 *Sequence of primitives in an NS user rejection of an NC establishment attempt*

An NS user may reject an NC establishment attempt by issuing an N-DISCONNECT request. The originator parameter in the N-DISCONNECT primitives will indicate NS user invoked NC release. The sequence of events is defined in the time sequence diagram in Figure 11/X.213.

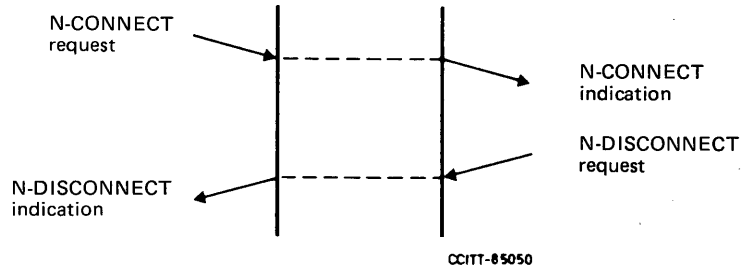


FIGURE 11/X.213

Sequence of primitives in NS user rejection of an NC establishment attempt

13.5 *Sequence of primitives in an NS provider rejection of an NC establishment attempt*

If the NS provider is unable to establish an NC, it indicates this to the requestor by issuing an N-DISCONNECT indication. The originator parameter in this primitive indicates an NS provider invoked NC release. The sequence of events is defined in the time sequence diagram in Figure 12/X.213.

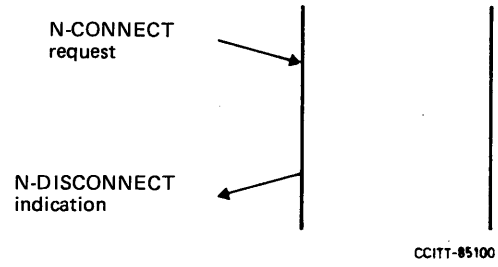


FIGURE 12/X.213

Sequence of primitives in NS provider rejection of an NC establishment attempt

14 Data transfer phase

14.1 *Data transfer*

14.1.1 *Function*

The data transfer service primitives provide for an exchange of NS-user-data called Network-Service-data-units (NSDUs), in either direction or in both directions simultaneously on an NC. The Network Service preserves both the sequence and the boundaries of the NSDUs.

Note – Designers of higher layer protocols using the Network Service should realize that the requested QOS applies to complete NSDUs, and that divisions of available NS-user-data into small NSDUs may have cost implications because of the impact on cost optimization mechanisms operated by the NS provider.

14.1.2 *Types of primitives and parameters*

Table 14/X.213 indicates the types of primitives and the parameters needed for data transfer.

TABLE 14/X.213

Data transfer primitives and parameters

Primitive / Parameter	N-DATA request	N-DATA indication
NS-user-data	X	X(=)
Confirmation request	X(C)	X(C=)

14.1.2.1 *NS-user-data parameter*

The NS-user-data parameter allows the transfer of an NSDU between NS users, without modification by the NS provider. The NS user may send any integer number of octets, one or greater, of NS-user-data that forms the NSDU.

14.1.2.2 *Confirmation request parameter*

The receipt confirmation of an NSDU transferred by means of an N-DATA primitive can be requested by setting the confirmation request parameter on the N-DATA request. The confirmation of receipt (COR) is provided by the N-DATA-ACKNOWLEDGE primitives (see § 14.2). The value of the confirmation request parameter may indicate either that a COR is requested or that it is not requested. The parameter may only be present if use of the receipt confirmation service was agreed by both NS users and the NS provider during the establishment of the NC.

14.1.3 *Sequence of primitives*

The operation of the Network Service in transferring NSDUs can be modelled as a queue of unknown size within the NS provider (see § 9). The ability of an NS user to issue an N-DATA request or of the NS provider to issue an N-DATA indication depends on the behaviour of the receiving NS user and the resulting state of the queue.

The sequence of primitives in a successful data transfer is defined in the time sequence diagram in Figure 13/X.213.

The sequence of primitives in Figure 13/X.213 may remain uncompleted if an N-RESET or an N-DISCONNECT primitive occurs.

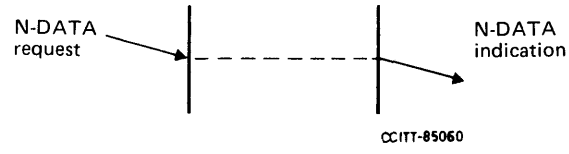


FIGURE 13/X.213

Sequence of primitives in data transfer

14.2 *Receipt confirmation service*

14.2.1 *Function*

The receipt confirmation service is requested by the confirmation request parameter on the N-DATA primitives. For each and every NSDU transferred with the confirmation request parameter set, the receiving NS user should return a confirmation of receipt (COR) by issuing an N-DATA-ACKNOWLEDGE request. Such CORs should be issued in the same sequence as the corresponding N-DATA indications were received, and will be conveyed by the NS provider so as to preserve them distinct from any previous or subsequent CORs. The NS user may thus correlate them with the original N-DATA primitives (with "confirmation requests" set) by counting.

N-DATA-ACKNOWLEDGE requests will not be subject to the flow control affecting N-DATA requests at the same NC endpoint; N-DATA-ACKNOWLEDGE indications will not be subject to the flow control affecting N-DATA indications at the same NC endpoint.

The use of the receipt confirmation service shall be agreed by the two NS users of the NC and the NS provider during NC establishment by use of the receipt confirmation selection parameter on the N-CONNECT primitives. The service need not be provided by all NS providers.

14.2.2 *Types of primitives and parameters*

The receipt confirmation service involves two primitives:

- N-DATA-ACKNOWLEDGE request;
- N-DATA-ACKNOWLEDGE indication.

These primitives do not convey any parameters.

14.2.3 *Sequence of primitives*

The sequence of primitives in a successful data transfer with receipt confirmation is defined in the time sequence diagram in Figure 14/X.213.

The sequence of primitives in Figure 14/X.213 may remain uncompleted if an N-RESET or an N-DISCONNECT primitive occurs.

An NS user must not issue an N-DATA-ACKNOWLEDGE request if no N-DATA indication with "confirmation request" set has been received or if a COR has already been issued for all such N-DATA indications. Following a reset procedure, signalled by means of an N-RESET indication or N-RESET confirm, an NS user must not issue an N-DATA-ACKNOWLEDGE request in response to an N-DATA indication (with "confirmation request" set) which was received before the reset procedure was signalled.

Note 1 – The witholding of COR by an NS user can have an effect on the throughput attainable on the NC.

Note 2 – The use of receipt confirmation on an NC may have an effect on the flow control of normal data on the NC. For example, the issuing of a COR may result in the relaxation of flow control on NS-user-data flowing in the opposite direction from the COR.

Note 3 – Receipt confirmation is included in the Network Service only to support existing features of Recommendation X.25.

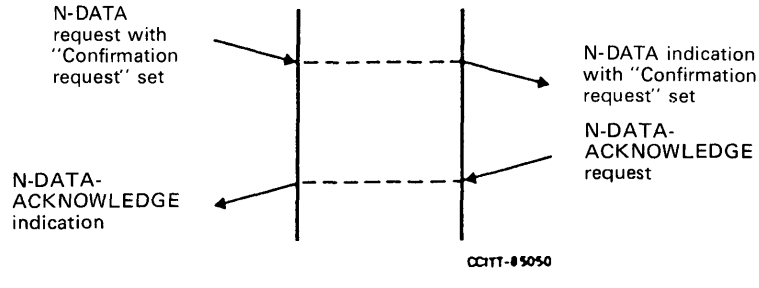


FIGURE 14/X.213

Sequence of primitives in successful data transfer with receipt confirmation

14.3 *Expedited data transfer service*

14.3.1 *Function*

The expedited data transfer service provides a further means of information exchange on an NC in both directions simultaneously. The transfer of expedited Network-Service-data-units (ENSDUs) is subject to different QOS and separate flow control from that applying to NS-user-data of the data transfer service. It is not intended to provide a qualified data transfer facility.

The NS preserves both the sequence and boundaries of the ENSDUs. The NS provider guarantees that an ENSDU will not be delivered after any subsequently issued NSDU or ENSDU on that NC.

The relationship between normal and expedited NS-user-data is modelled by the operation of changing of order within queues as described in § 9.2.3. In particular, expedited NS-user-data can still be delivered when the receiving NS user is not accepting normal NS-user-data. However, the amount of normal NS-user-data bypassed by such changing of order cannot be predicted or guaranteed. Expedited data transfer cannot be guaranteed to bypass blockages in normal data flow where these blockages are occurring in lower layers.

The expedited data transfer service is a provider-option which may not be available in the Network Service. Its use must be agreed to by the two NS users of the NC and the NS provider during NC establishment by means of the expedited data selection parameter on the N-CONNECT primitives (see § 12.2.6).

14.3.2 *Types of primitives and parameters*

Table 15/X.213 indicates the types of primitives and the parameters needed for expedited data transfer.

TABLE 15/X.213

Expedited Data Transfer primitives and parameters

Parameter \ Primitive	N-EXPEDITED-DATA request	N-EXPEDITED-DATA indication
NS-user-data	X	X(=)

14.3.2.1 *NS-user-data parameter*

The NS-user-data parameter allows the transfer of expedited NS-user-data between NS users, without modification by the NS provider. The NS user may send any integer number of octets of expedited NS-user-data between 1 and 32 inclusive.

14.3.3 *Sequence of primitives*

The sequence of primitives in a successful expedited data transfer is defined in the time sequence diagram in Figure 15/X.213.

The sequence of primitives in Figure 15/X.213 may remain uncompleted if an N-RESET or an N-DISCONNECT primitive occurs.

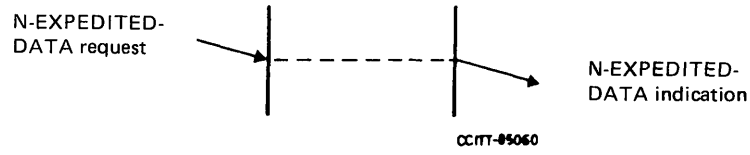


FIGURE 15/X.213

Sequence of primitives in expedited data transfer

14.4 *Reset service*

14.4.1 *Function*

The reset service may be used:

a) by the NS user to resynchronize the use of the NC; or

b) by the NS provider to report detected loss of NS-user-data unrecoverable within the NS provider. All loss of NS-user-data which does not involve loss of the NC is reported in this way.

Invocation of the reset service will unblock the flow of NSDUs and ENSDUs in case of congestion of the NC; it will cause the NS provider to discard NSDUs, ENSDUs, or CORs associated with the NC, and to notify any NS user or users that did not invoke reset that a reset has occurred. The service will be completed in a finite time, irrespective of the acceptance of NSDUs, ENSDUs, and CORs by the NS users. Any NSDUs, ENSDUs, or CORs not delivered to the NS users before completion of the service will be discarded by the NS provider.

14.4.2 *Types of primitives and parameters*

Table 16/X.213 indicates the types of primitives and the parameters needed for the reset service.

TABLE 16/X.213

Reset primitives and parameters

Parameter \ Primitive	N-RESET request	N-RESET indication	N-RESET response	N-RESET confirm
Originator		X		
Reason	X	X		

14.4.2.1 *Originator parameter*

The originator parameter indicates the source of the reset. Its value indicates either the "NS user", "NS provider", or "undefined".

14.4.2.2 *Reason parameter*

The reason parameter gives information indicating the cause of the reset. The value conveyed in this parameter will be as follows:

a) When the originator parameter indicates an NS provider invoked reset, the value is one of:

   i) "congestion";

   ii) "reason unspecified".

b) When the originator parameter indicates an NS user invoked reset, the value is "user resynchronization".

c) When the originator parameter has the value "undefined", then the value of the reason parameter is also "undefined".

14.4.3 *Sequence of primitives*

The interactions between each NS user and the NS provider will be an exchange of these primitives, namely either:

a) an N-RESET request from the NS user, followed by an N-RESET confirm from the NS provider; or

b) an N-RESET indication from the NS provider, followed by an N-RESET response from the NS user.

The N-RESET request acts as a synchronization mark in the stream of NSDUs, ENSDUs, and CORs transmitted by the issuing NS user. The N-RESET indication likewise acts as a synchronization mark in the stream of NSDUs, ENSDUs, and CORs received by the receiving NS user. Similarly, the N-RESET response acts as a synchronizing mark in the stream of NSDUs, ENSDUs, and CORs transmitted by the respoding NS user, while the N-RESET confirm acts as a synchronization mark in the stream of NSDUs, ENSDUs, and CORs received by the NS user which originally invoked the Reset.

The resynchronizing properties of the reset service are that:

1) No NSDU, ENSDU, or COR transmitted by the NS user *before* the synchronization mark in that transmitted stream will be delivered to the other NS user *after* the synchronization mark in that received stream.

   The NS provider will discard all NSDUs, ENSDUs, and CORs submitted before the issuing of the N-RESET request which have not been delivered to the receiving NS user when the NS provider issues the N-RESET indication.

   Also, the NS provider will discard all NSDUs, ENSDUs, and CORs submitted before the issuing of the N-RESET response which have not been delivered to the initiator of the N-RESET when the NS provider issues the N-RESET confirm.

2) No NSDU, ENSDU, or COR transmitted by an NS user *after* the synchronization mark in that transmitted stream will be delivered to the other NS user *before* the synchronization mark in that received stream.

The N-RESET confirm may be issued to the initiator of the reset before the N-RESET indication is issued to the other NS user. The complete sequence of primitive depends upon the origin of the reset action and the occurrence or otherwise of resets with conflicting origins. Thus the reset service may be:

i) invoked by one NS user, lading to interaction a) with that NS user and interaction b) with the peer NS user;

ii) invoked by both NS users, leading to interaction a) with both NS users;

iii) invoked by the NS provider, leading to interaction b) with both NS users;

iv) invoked by one NS user and the NS provider, leading to interaction a) with the originating NS user and b) with the peer NS user.

The sequence of primitives in these four cases is defined in the time sequence diagrams in Figures 16/X.213 to 19/X.213.

Further, there may be circumstances of reset "collision" which result in the number of reset procedures observed at one NC endpoint being different from the number of reset procedures observed at the other NC endpoint. Such circumstances result in two additional cases which may occur, where the reset service may be:

v) invoked by one NS user while a previous reset procedure is still incomplete at the other NS user, leading to additional interaction a) with the NS user invoking the subsequent reset, only;

vi) invoked by the NS provider at one NC endpoint, while a previous reset procedure is still incomplete at the other, leading to additional interaction b) with the NS user at the first NC endpoint, only.

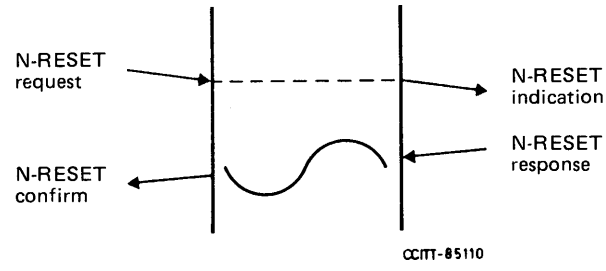


FIGURE 16/X.213

Sequence of primitives in NS user invoked reset

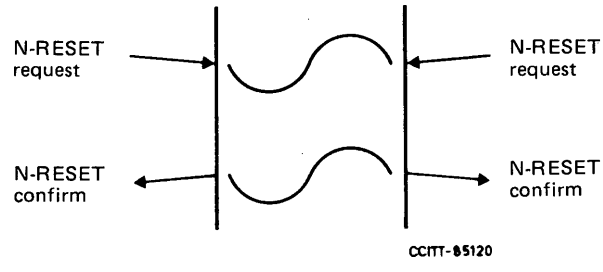


FIGURE 17/X.213

Sequence of primitives in simultaneous NS user invoked reset

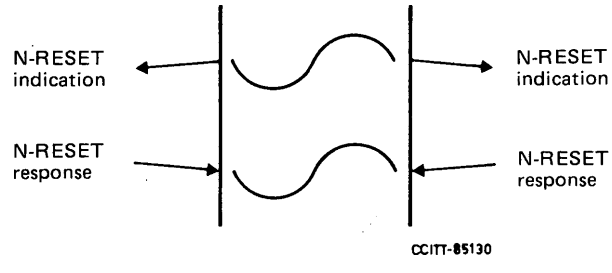


FIGURE 18/X.213

Sequence of primitives in NS provider invoked reset

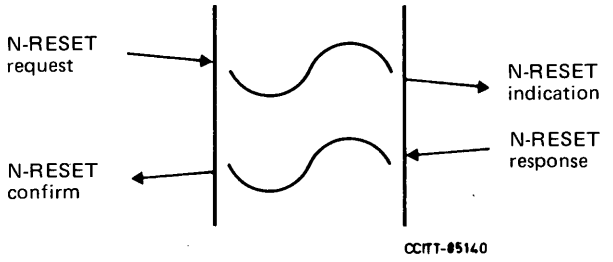


FIGURE 19/X.213

Sequence of primitives in simultaneous NS user and NS provider invoked reset

There are many possible sequences of reset primitives for the two NC endpoints which may occur for cases v) and vi). These are not illustrated here by time sequence diagrams, but may be derived using the constraints on the allowed sequence of primitives for each NC endpoint, and the reset sequences illustrated in Figures 16/X.213 to 19/X.213. The synchronizing properties associated with the issuance of the N-RESET primitives are the same for all of the six cases outlined.

Note – Situations in which the number of reset procedures at the two ends of a NC which are not the same are not described by the operation of the queue model in § 9.2.

Any sequence of reset primitives may remain uncompleted if an N-DISCONNECT primitive occurs. Once a reset procedure has been invoked at an NC endpoint (by means of an N-RESET request or N-RESET indication primitive), no further N-DATA, N-EXPEDITED-DATA, or N-DATA-ACKNOWLEDGE primitive can be issued by either the NS user or the NS provider until the reset procedure has completed (by means of an N-RESET confirm or N-RESET response).

ANNEX A

(to Recommendation X.213)

Network Layer Addressing

A.0 *Introduction*

This annex defines the abstract syntax and semantics of the Network address (Network service access point address). The Network address is the address that appears in the primitives of the connection-mode Network service as the *calling address*, *called address* and *responding address* parameters, and in the primitives of the connectionless-mode Network service as the *source address* and *destination address* parameters.

A.1 *Scope and field of application*

The scope of this annex is the definition of the abstract syntax and semantics of the Network address. This annex does not specify the way in which the semantics of the Network address are encoded in Network layer protocols. The field of application of this annex is the same as the field of application described in section 1 of X.213.

A.2 *References*

ISO 646 Information processing – ISO 7-bit coded character set for information interchange,

ISO 2375 Data processing – Procedure for registration of escape sequences,

ISO 3166 Codes for the representation of names of countries,

ISO 6523 Data interchange – Structure for the identification of organizations,

E.163 Numbering plan for the international telephone service,

E.164 The numbering plan for the ISDN era,

F.69 Plan for telex destination codes,

X.121 International numbering plan for public data networks,

X.200 Reference model for open systems interconnection for CCITT applications [see also ISO 7498],

X.210 OSI layer service conventions [see also ISO TR 8509],

X.300 General principles for interworking between public networks and between public networks and other networks for the provision of data transmission services.

A.3 *Definitions*

A.3.1 *Reference model definitions*

This annex makes use of the following terms defined in X.200. Terms that are defined in X.200 using the generic prefix "(N)" appear in this annex with the layer-specific prefix "Network".

(N) – layer,

(N) – service,

(N) – service access point,

(N) – service access point address,

(N) – entity,
routing,

(N) – address,

(N) – protocol control information,

(N) – protocol data unit,
OSI environment,
title,

(N) – relay.

A.3.2 *Service conventions definitions*

This annex makes use of the following terms defined in X.210.

– service user,

– service provider,

– service primitive,

– indication (primitive).

A.3.3 *Network layer architecture definitions*

This annex makes use of the following terms defined in X.300.

– subnetwork,

– real subnetwork,

– subnetwork service,

– real end system,

– interworking unit,

– intermediate system,

– relay entity.

A.3.4 *Network addressing definitions*

This annex makes use of the following terms as defined below:

A.3.4.1 **DTE address**: information used to identify a point of attachment to a public data network.

A.3.4.2 **subnetwork point of attachment**: a point at which a real end system, interworking unit, or real subnetwork is attached to a real subnetwork, and a conceptual point at which a subnetwork service is offered within an end or intemediate system.

A.3.4.3 **subnetwork point of attachment address**: information used in the context of a particular real subnetwork to identify a subnetwork point of attachment; or information used in the context of a particular subnetwork to identify the conceptual point within an end or intermediate system at which the subnetwork service is offered. This term is used interchangeably with the (equivalent) shortened form *subnetwork address.*

A.3.4.4 **network protocol address information**: information encoded in a Network protocol data unit to carry the semantics of a Network service access point address. (This is known as an "address signal" or as the "coding of an address signal" in the public network environment.)

A.3.4.5 **naming domain**: a context within which a name allocated by a naming authority is unambiguous. Where the name is an address, the context within which the name is allocated is called an *addressing domain.*

A.3.4.6 **global network addressing domain**: an addressing domain consisting of all of the Network service access point addresses in the OSI environment.

A.3.4.7 **network addressing domain**: a subset of the global network addressing domain consisting of all of the Network service access point addresses allocated by one or more addressing authorities.

A.3.4.8 **naming authority**: that which allocated names from a specified naming domain, and which ensures that names so allocated are unambiguous. Where the naming authority allocates addresses, it is called an *addressing authority.*

A.3.4.9 **network addressing authority**: an addressing authority that assigns and administers Network service access point addresses within one or more network addressing domains.

A.3.4.10 **abstract syntax**: a notation which enables data types to be defined, and values of those types specified, without determining the way in which they will be represented (encoded) for transfer by protocols.

A.4 *Abbreviations*

This annex makes use of the following abbreviations:

NSAP Network service access point,
NPAI Network protocol addressing information,
DCC data country code,
CC country code,
ICD international code designator,
PSTN public switched telephone network,
ISDN integrated services digital network,
IDP initial domain part,
AFI authority and format identifier,
IDI initial domain identifier,
DSP domain specific part,
NPDU Network protocol data unit,
SNPA subnetwork point of attachment,
PTT postal, telephone and telegraph,
DRPF decimal reference publication format,
HRPF hexadecimal reference publication format.

A.5 *Conventions*

No particular standard conventions are invoked by this annex..

A.6 *Concepts and terminology*

A.6.1 *Network addresses*

This annex defines the Network service access point (NSAP) address. Since the term "network address" is commonly used in different contexts to refer to different things, a more specific description of this concept is introduced below.

A.6.1.1 *Subnetwork address*

In one context, the term "network address" may be used to refer to the point at which a *real end system*, *real subnetwork*, or *interworking unit* is attached to a real subnetwork, or to the point at which the subnetwork service is offered within an end or intermediate system. In the case of attachment to a public data network, this point is called a DTE/DCE interface, and the term "DTE address" is used in reference to it.

The specific term *subnetwork address* (or *subnetwork point of attachment address*) is used in this case, as illustrated in Figure A-1/X.213.

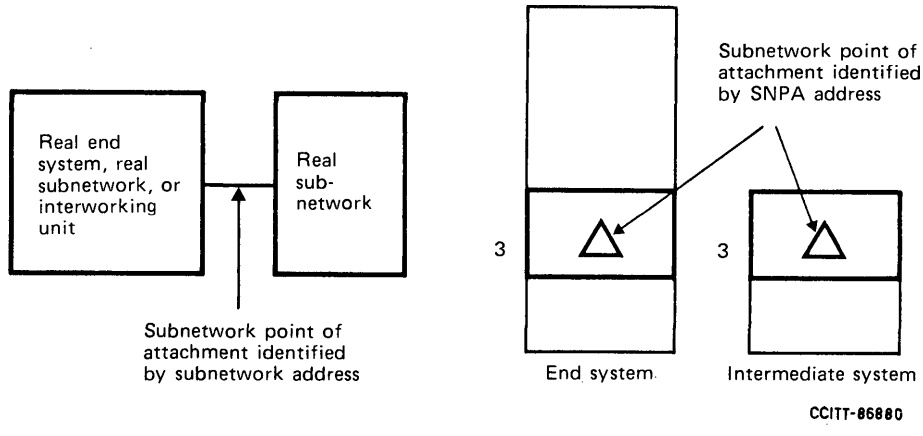


FIGURE A-1/X.213

Subnetwork address

The subnetwork address is the information that a real subnetwork needs to identify a particular real end system, another real subnetwork, or an interworking unit, which is attached to that real subnetwork.

In the public network environment, the subnetwork address is what the public network operates on.

Note – The point identified by a subnetwork address is a point of interconnection between a real end system or interworking unit and a real subnetwork (in particular, in a public data network environment, a DTE/DCE interface), and is not a Network service access point.

A.6.1.2 *NSAP address*

In another context, the term "network address" is used to refer to the *Network service access point* (NSAP) at which the OSI Network service is made available to a Network service user by the Network service provider.

The specific term *NSAP address* is used in this case, as illustrated in Figure A-2/X.213:

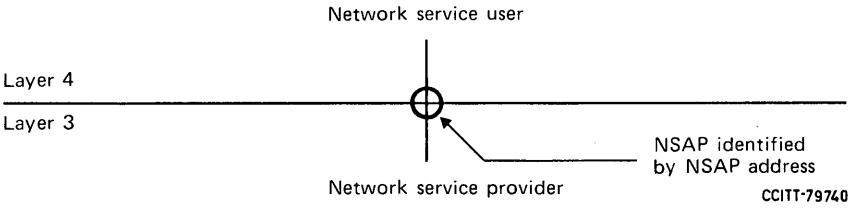


FIGURE A-2/X.213

NSAP address

The NSAP address is the information that the OSI Network service provider needs to identify a particular Network service access point. The values of the *called address*, *calling address*, and *responding address* parameters of the N-CONNECT primitive, of the *responding address* parameter of the N-DISCONNECT primitive, and of the *source address* and *destination address* parameters of the N-UNITDATA primitive, are NSAP addresses.

It should be noted that since the Network service primitives are conceptual, no particular encoding of the NSAP address is specified by the Network service definition.

In both CCITT and ISO usage, the terms "Network Address" (with both the N and the A printed in capital letters) and "global network address" are synonymous with the term "NSAP address". Use of the term "NSAP address" is preferred when it is essential to avoid confusion, particularly in spoken references in which "capitalization" is not possible.

A.6.1.3 *Network protocol address information*

In a third context, the term "network address" is used to refer to an address that is carried as *Network protocol control information* in a Network protocol data unit (NPDU).

The specific term *Network protocol address information* (NPAI) is used in this case.

In the public network environment, NPAI is also known as an "address signal" or as the "coding of an address signal".

There is a relationship between the NSAP address that appears in Network service primitives and the NPAI that appears in a Network layer protocol, in that the semantics of the NSAP address are preserved by the NPAI. The precise encoding of NPAI is defined by Network layer protocol standards, which also specify the relationship between the NSAP address and the NPAI encoding employed by the protocol.

A.6.2 *Domains*

A.6.2.1 *Global network addressing domain*

The *global network addressing domain* is an addressing domain consisting of all of the NSAP addresses in the OSI environment.

A.6.2.2 *Network addressing domain*

A *network addressing domain* is a subset of the global network addressing domain consisting of all of the NSAP addresses that are assigned by one or more addressing authorities. Every NSAP address is part of a network addressing domain that is administered directly by one and only only addressing authority. If that network addressing domain is part of a (hierarchically) higher addressing domain (which must wholly contain it), the authority for the (hierarchically) lower domain is authorized by the authority for the higher domain to assign NSAP addresses from the lower domain. All network addressing domains are in this way ultimately part of the global network addressing domain, for which the addressing authority is this annex.

The relationship of the concepts of A.6.2.1 and A.6.2.2 is illustrated by Figure A-3/X.213:

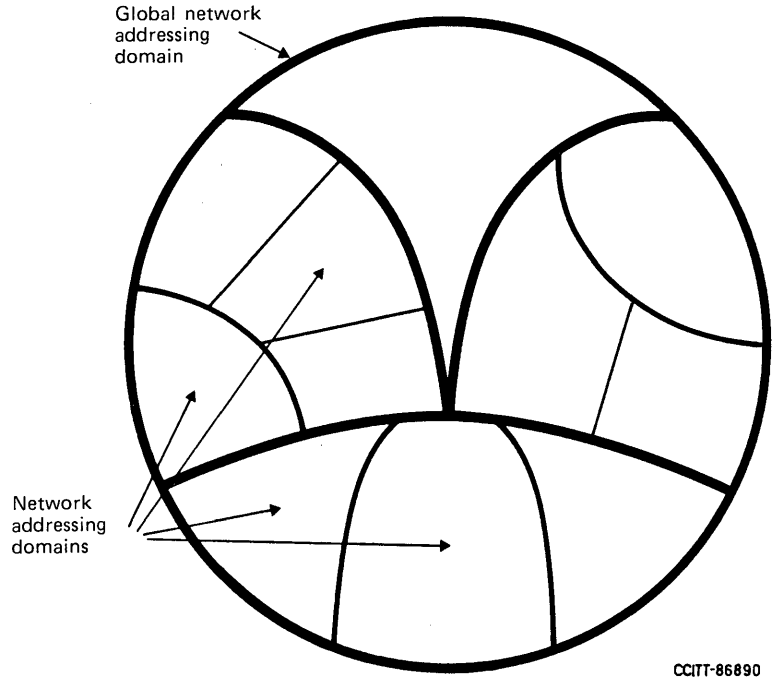


FIGURE A-3/X.213

Network addressing domains

A.6.3 *Authorities*

The uniqueness of identifiers within a network addressing domain is ensured by an *authority* associated with that domain. The term "authority" does not necessarily refer to an organization or administration; it is intended to refer to whatever it is (in an abstract sense) that ensures the uniqueness of identifiers in the associated domain.

A network addressing domain is characterized by the addressing authority that administers the domain and by the rules that are established by that authority for specifying identifiers and identifying subdomains. The authority responsible for each domain determines how identifiers will be assigned and interpreted within that domain, and how any further subdomains will be created.

The operation of an authority is independent of that of other authorities at the same level of the hierarchy, subject only to any common rules imposed by the parent authority.

A.6.4 *Network address allocation*

An addressing authority shall either allocate complete NSAP addresses, or authorize one or more other authorities to allocate addresses. Each address allocated by an addressing authority shall include a domain identifier which identifies the allocating authority. An address shall not be allocated to identify a domain or NSAP if the address had previously been allocated in some other domain or NSAP, unless the authority can ensure that all use of the previous allocation has ceased.

The authority shall ensure that allocations are made in such a way that efficient use is made of the address space.

A.7 *Principles for creating the OSI Network addressing scheme*

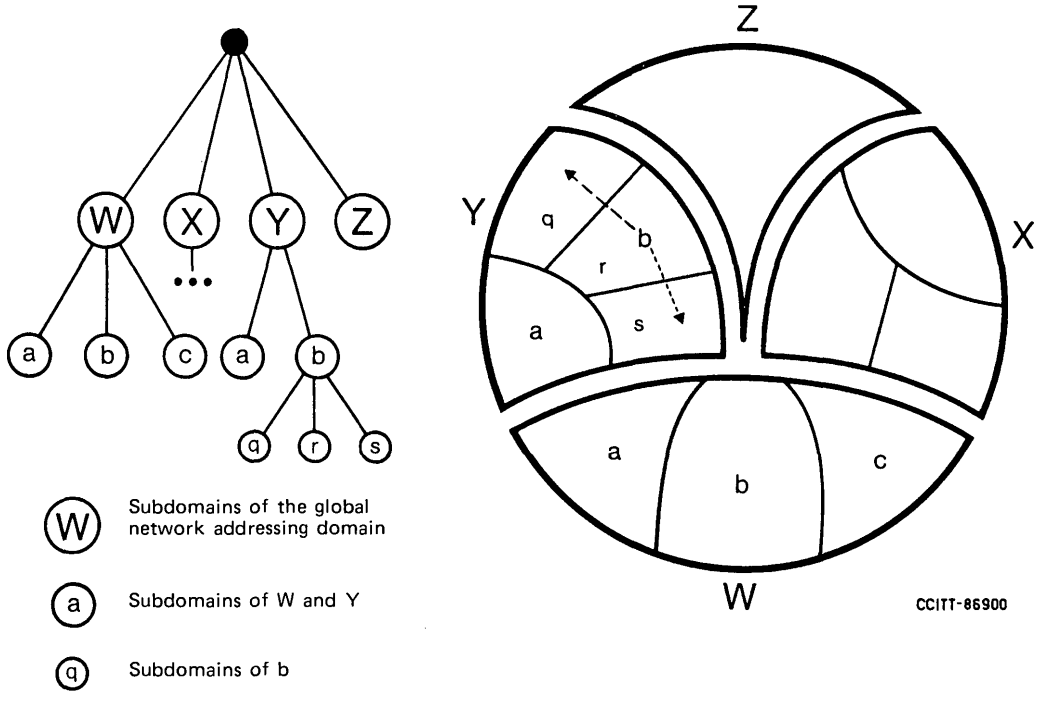
A.7.1 *Hierarchical structure*

NSAP addresses are based on the concept of hierarchical addressing domains, as explained in A.6. Each domain may be further partitioned into subdomains. Accordingly, NSAP addresses have a hierarchical structure.

The conceptual structure of NSAP addresses follows the principle that, at any level of the hierarchy, an initial part of the address unambiguously identifies as subdomain, and the rest is allocated by the authority associated with the subdomain to unambigously identify either a lower level subdomain or an NSAP within the subdomain. The part of the address that identifies the subdomain depends on the level at which the address is viewed.

Note – This conceptual structure should not be considered as implying any detailed administration of NSAP addresses.

Graphical representation of the hierarchical structure of NSAP addresses may be made according to an inverted tree diagram, as in Figure A-4/X.213 (a), or a domain diagram, as in Figure A-4/X.213 (b):



(a) Inverted tree diagram

(b) Domain diagram

FIGURE A-4/X.213

Hierachical structure of NSAP addresses

A.7.2 *Global identification of any NSAP*

In the context of Open Systems Interconnection, it is possible to identify any NSAP within the global network addressing domain (see A.6.2.1). Consequently,

a) an NSAP address can be defined to unambiguously identify any NSAP;

b) at any NSAP, it is possible to identify any other NSAP, within any OSI end system;

c) the Network layer protocols established between correspondent Network entities convey the complete semantics of an NSAP address (see A.6.1.3);

d) an NSAP address always identifies the same NSAP, regardless of which Network service user enunciates the address; and

e) a Network service user, when given an NSAP address by the Network service provider in an Indication service primitive, may subsequently use that NSAP address in another instance of communication with the corresponding NSAP.

Note – The global identification of NSAPs does not imply the universal availability of directory functions required to enable communication among all NSAPs to which NSAP addresses have been assigned, nor does it prevent the imposition of external restrictions on communication based on the technical feasibility of interconnection, security, charging, etc.

A.7.3 *Route independence*

Network service users cannot derive routing information from an NSAP address. They cannot influence the Network service provider's choice of route by means of the source and destination NSAP addresses. Similarly, they cannot determine the route that was used by the Network service provider by examining the source and destination NSAP addresses. This is not intended to exclude the possibility that an OSI end system may need to influence the route selected for a particular instance of communication with another OSI end system. (In particular, it may need to influence the selection of intermediate systems to be used, and the paths to be taken between them.) The means whereby such an influence may be exerted is, however, *not* the NSAP address. Elements of Network layer protocol may be required to control routing within intermediate systems; such elements of protocol are distinct from the network protocol address information (NPAI).

Notwithstanding the restrictions imposed on the use that a Network service *user* may make of an NSAP address, it is recognized that NSAP addresses should be constructed in such a way that routing through interconnected subnetworks is facilitated. That is, space the Network service *provider*, and relay-entities in particular, may take advantage of the address structure to achieve economical processing of routing aspects.

A.8 *Network address definition*

The intent of this annex is best served by maintaining clear distinctions among three concepts: the *abstract semantics* of the NSAP address; the *abstract syntax* employed in this annex as a means of defining the abstract semantics of the NSAP address, and employed by network addressing authorities as a means of allocating and assigning NSAP addresses; and the *encoding* of the NSAP address semantics as NPAI in Network layer protocols. These distinctions are illustrated in Figure A-5/X.213:

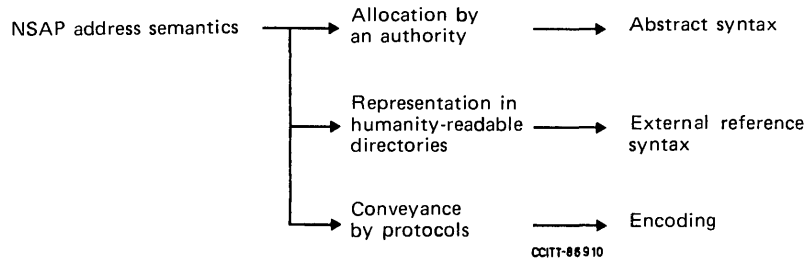


FIGURE A-5/X.213

Relationship of NSAP address semantics and syntax

This annex does not specify the way in which the semantics of the NSAP address are encoded in Network layer protocols, although preferred encodings are defined in A.8.3. Network layer protocol specifications define the way in which the NSAP address is encoded as NPAI (see A.6.1.3).

A.8.1 *Network address semantics*

The NSAP address consists of two basic semantic parts. The first part is the *initial domain part* (IDP). The second part is the *domain specific part* (DSP). This is illustrated by Figure A-6/X.213.

Following the conceptual structure of NSAP addresses described in A.7.1, the IDP is a network addressing domain identifier: it specifies a subdomain of the global network addressing domain (see Figure A-4/X.213), and identifies the network addressing authority responsible for assigning NSAP addresses in the specified subdomain. The DSP is the corresponding subdomain address. A further substructure of the DSP may or may not be defined by the authority identified by the IDP.

A.8.1.1 *The IDP*

The initial domain part of the NSAP address itself consists of two parts. The first part is the *authority and format identifier* (AFI). The second part is the *initial domain identifier* (IDI). This is illustrated by Figure A-6/X.213:

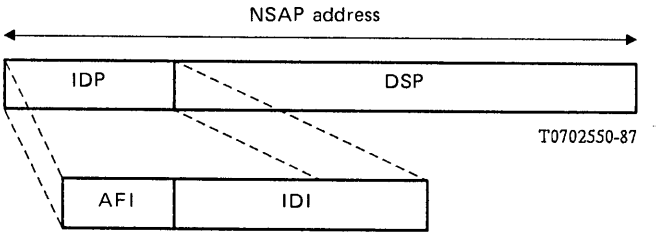


FIGURE A-6/X.213

NSAP address structure

A.8.1.1.1 *The AFI*

The authority and format identifier specifies:

a) the format of the IDI (see A.8.2.1.2);
b) the network addressing authority responsible for allocating values of the IDI (see A.8.2.1.2);
c) whether or not leading zero digits in the IDI are significant (see A.8.3); and
d) the abstract syntax of the DSP (see A.8.2.2 and A.8.2.3).

A.8.1.1.2 *The IDI*

The initial domain identifier specifies:

a) the network addressing domain from which values of the DSP are allocated; and
b) the network addressing authority responsible for allocating values of the DSP from that domain.

A.8.1.2 *The DSP*

The semantics of the DSP are determined by the network addressing authority identified by the IDI (see A.8.1.1.2).

A.8.2 *Network address abstract syntax*

The Network address is defined in this annex in terms of an abstract syntax in which the semantics of the Network address can be expressed. The use of this abstract syntax as a descriptive device enables this annex to convey, in written form, a complete definition of the Network address without restricting it to the specific encoding of the NPAI. It also enables this annex to identify two alternative preferred encodings of the Network address, to which reference may be made by Network layer protocol specifications so as to unambiguously define the way in which the Network address is encoded as NPAI.

A.8.2.1 *Abstract syntax and allocation of the IDP*

This section defines the abstract syntax of the AFI, the currently allocated values of the AFI, and the IDI formats corresponding to the allocated AFI values. Among the currently allocated values of the AFI are values reserved for assignment to new IDI formats which may be identified by ISO or CCITT. Assignment of these AFI values to new IDI formats by either ISO or CCITT must be accompanied by appropriate modification of this annex. Allocation of new AFI values shall be by joint agreement between ISO and CCITT, and will require an appropriate modification of this annex.

The abstract syntax of the IDP is decimal digits. The allocation of the AFI ensures that the first decimal digit of the IDP can never be zero (see A.8.1.1). This provides an escape mechanism for use by protocols that expect to hold incomplete NSAP addresses in a field that normally carries a complete NSAP address. When the NSAP address is represented as binary octets, the representation of the IDP is as defined in A.8.3.1.

The length of the IDP depends on the IDI format specified by the value of the AFI. The IDP length associated with each IDI format is given in A.8.2.1.2.

A.8.2.1.1 *Abstract syntax and allocation of the AFI*

The AFI consists of an integer with a value between 0 and 99 with an abstract syntax of two decimal digits. The values of the AFI are allocated or reserved as shown in Table A-1/X.213.

A.8.2.1.2 *Format and allocation of the IDI*

A specific combination of IDI format and DSP abstract syntax is associated with each allocated AFI value, as summarized in Table A-2/X.213. Two AFI values are associated with each combination that involves a variable-length IDI format. In each case, both of the AFI values identify the same combination of IDI format and DSP abstract syntax. The numerically lower AFI value is used when the first significant digit in the IDI is nonzero. The numerically greater AFI value is used when the first significant digit in the IDI is zero.

In the following subsections, references are made to the number used in particular subnetwork numbering plans, and to the entity identified by such a number. These references refer to the entity located at the subnetwork point of attachment specified by the number, and not to some other entity (for example, a PTT Administration) whose identity might be inferred from inspection of some part of the number. The authority in such cases is therefore the authority associated with the entity at the SNPA, and is identified by the *full* number.

TABLE A-1/X.213

AFI allocations

00-09	Reserved – will not be allocated
10-35	Reserved for future allocation by joint agreement of ISO and CCITT
36-59	Allocated and assigned to the IDI formats defined in § A.8.2.1.2
60-69	Allocated for assignment to new IDI formats by ISO
70-79	Allocated for assignment to new IDI formats by CCITT
80-99	Reserved for future allocation by joint agreement of ISO and CCITT

TABLE A-2/X.213

Allocated AFI values

DSP syntax / IDI format	Decimal	Binary	Character (ISO 646)	National Character
X.121	36, 52	37, 53	////////////////	////////////////
ISO DCC	38	39	////////////////	////////////////
F.69	40, 54	41, 55	////////////////	////////////////
E.163	42, 56	43, 57	////////////////	////////////////
E.164	44, 58	45, 59	////////////////	////////////////
ISO 6523-ICD	46	47	////////////////	////////////////
Local	48	49	50	51

Note – The Local IDI format is provided to accommodate the coexistence of OSI and non-OSI network addressing schemes, particularly in the context of a transition from non-OSI to OSI protocols. To provide the greatest flexibility in these environments, character and national character DSP syntaxes are defined for the Local IDI format.

A.8.2.1.2.1 *X.121 IDI format*

The IDI consists of a sequence of up to 14 digits allocated according to CCITT Recommendation X.121. The full X.121 number identifies an authority responsible for allocating and assigning values of the DSP.

IDP length: up to 16 digits.

A.8.2.1.2.2 *ISO DCC IDI format*

The IDI consists of a three-digit numeric code allocated according to ISO 3166. For countries with an ISO member body, the code is assigned to the ISO member body in the country identified by the code. For countries with no ISO member body, the code is assigned to an appropriately sponsored organization in the country identified by the code. The DSP is allocated and assigned by the ISO member body or sponsored organization to which the ISO DCC value has been assigned, or by an organization designated by the holder of the ISO DCC value to carry out this responsibility.

IDP length: 5 digits.

A.8.2.1.2.3 *F.69 IDI format*

The IDI consists of a telex number of up to 8 digits, allocated according to CCITT Recommendation F.69, commencing with a 2- or 3-digit destination code. The full telex number identifies an authority responsible for allocating and assigning values of the DSP.

IDP length: Up to 10 digits.

A.8.2.1.2.4 *E.163 IDI format*

The IDI consists of a public switched telephone network (PSTN) number of up to 12 digits allocated according to CCITT Recommendation E.163, commencing with the PSTN country code. The full PSTN number identifies an authority responsible for allocating and assigning values of the DSP.

IDP length: up to 14 digits.

A.8.2.1.2.5 *E.164 IDI format*

The IDI consists of an ISDN number of up to 15 digits allocated according to CCITT Recommendation E.164, commencing with the ISDN country code. The full ISDN number identifies an authority responsible for allocating and assigning values of the DSP.

IDP length: Up to 17 digits.

A.8.2.1.2.6 *ISO 6523-ICD IDI format*

The IDI consists of a 4-digit International Code Designator (ICD) allocated according to ISO 6523. The ICD identifies an organizational authority responsible for allocating and assigning values of the DSP.

IDP length: 6 digits.

Note – The use of an ICD in this context is in addition to, and does not affect the uses identified in ISO 6523. Of the things specified by ISO 6523, *only* the ICD is relevant to this annex.

A.8.2.1.2.7 *Local IDI format*

The IDI is null.

IDP length: 2 digits.

Note 1 – The use of a particular IDI format as the basis for allocating an NSAP address does not constrain routing to that NSAP to go through any particular system or subnetwork. For example, the use of the E.163 IDI format as the basis for allocating an NSAP address does not mean that access to the NSAP with that address necessarily involves the user of the public telephone network (see A.7.3).

Note 2 – The IDI formats that are based on CCITT numbering plans may be affected by any changes to those plans. It should be understood that in identifying and describing these formats, this annex observes the current status of CCITT work on numbering plans, and does not establish any preference or position concerning the way in which CCITT may choose to modify the plans, or their relationships with one another, in the future. Changes to this annex may be necessary to take any such further work by CCITT into account. For example, the CCITT numbering plans in some cases may provide escape mechanisms (such as a zero, 8, or 9 prefix) from one numbering plan to another. This results in the possibility of a choice that must be made concerning which IDI format should be used for the allocation of NSAP addresses, and may also lead to suggestions that it is not necessary to include all of the IDI formats that are based on CCITT Recommendations in this annex. Such choices, however, are made within the context and responsibility of CCITT, and no preference for one choice or another is made or implied by this annex.

A.8.2.2 *Abstract syntax and allocation of the DSP*

Values of the DSP are allocated by the network addressing authority identified by the IDI in the syntax identified by the AFI (see A.8.1.1.2 and A.8.2.1.2). The allocating authority specifies the format and semantics of the DSP. If the authority identified by the IDI authorizes one or more authorities to allocate semantic parts of the DSP, then all of those authorities must allocate using the same abstract syntax used by the parent authority.

A network addressing authority may choose to allocate NSAP addresses with the DSP in a decimal or binary abstract syntax for all IDI formats. When the IDI format is "Local", an authority may, in addition, choose to allocate NSAP addresses with the DSP in a character (ISO 646) or National Character abstract syntax (see Table A-2/X.213 and § A.9). A network addressing authority may allocate NSAP addresses with no DSP (that is, addresses consisting of an IDP only) if and only if the value of the AFI specifies a *decimal* DSP syntax; a non-null DSP must be present for all other AFI values.

A.8.2.3 *Abstract syntax of the DSP*

The DSP may be allocated by the responsible authority in one of the following four syntaxes, depending on the value of the AFI:

a) *binary* – the DSP consists of one or more binary octets, up to the maximum specified in Table A-3/X.213,

b) *decimal* – the DSP, if present, consists of one or more decimal digits, up to the maximum specified in Table A-3/X.213,

c) *character* – the DSP consists of one or more of those ISO 646 graphic characters with no national variant, plus the space character, up to the maximum specified in Table A-3/X.213,

d) *national character* – The DSP consists of one or more characters from a national character set determined by the allocating authority, up to the maximum specified in Table A-3/X.213.

Table A-3/X.213 gives the maximum length of the DSP in its abstract syntax for each of the IDI formats defined in A.8.2.1.2. The corresponding total NSAP address lengths are given in A.8.4.

TABLE A-3/X.213

Maximum DSP length

DSP syntax / IDI format	Decimal digits	Binary octets	ISO 646 Characters	National Characters
X.121	24	9	////////////////	////////////////
ISO DCC	35	14	////////////////	////////////////
F.69	30	12	////////////////	////////////////
E.163	26	10	////////////////	////////////////
E.164	23	9	////////////////	////////////////
ISO 6523-ICD	34	13	////////////////	////////////////
Local	38	15	19	7

Note 1 – The values for the «Local» IDI format assume a National Character representation of one character as two binary octets (see A.8.3.1 and A.8.3.2).

Note 2 – These maximum values are dictated by the requirement that all entries in Table A-5/X.213 be less than or equal to 40 decimal digits or 20 binary octects.

A.8.3 *Network address encodings*

As described in A.8.1, the semantics of the NSAP address are represented by three fields in the following order:

a) the AFI, with an abstract syntax of two decimal digits;

b) the IDI, with an abstract syntax of a variable number of decimal digits; and

c) the DSP, with an abstract syntax of a variable number of one and only one of the following types; binary octets, decimal digits, characters, or national characters.

This annex does not specify the way in which the semantics of an NSAP address are encoded in Network layer protocols. These encodings are specified in Network layer protocol specifications.

Nevertheless, this annex identifies two alternative "preferred" encodings of the Network address (see A.8.3.1 and A.8.3.2). Reference to these encodings may be made by Network layer protocol specifications. It is possible that the encoding used to convey the Network address semantics as Network protocol addressing information (NPAI) in a Network layer protocol may be chosen to be identical to one of these preferred encodings. However, it is not required that this be the case (see A.9).

The entire NSAP address, taken as a whole, may be represented explicitly as a string of either *decimal digits* (decimal encoding) or *binary octets* (binary encoding) as defined below. Network layer protocol that specify the encoding of the Network address semantics by making reference to this annex must specify the way in which either the preferred decimal encoding or the preferred binary encoding is used to convey the Network address semantics as NPAI (see A.6.3.1).

Both of the preferred encodings identified in A.8.3.1 and A.8.3.2 require that the IDI be padded with non-significant leading pad digits whenever (a) the AFI specifies a variable-length IDI format, and (b) the value of the IDI is a string of decimal digits that is shorter than the maximum length of the IDI for that format (see A.8.2.1.2). This ensures that the end of the IDI (and thus of the IDP) can be determined; neither preferred encoding reserves an explicit syntactic marker for this purpose. It is necessary, in these cases, to make a distinction between significant and non-signigicant leading zero digits in the IDI, in order to ensure that non-significant pad digits are not confused with significant IDI digits. This distinction is provided, for each of the variable length IDI formats, by the allocation of two AFI values for each combination of IDI format and DSP abstract syntax (see A.8.2.1.1). In A.8.3.1 (b) and A.8.3.2 (b), the term "leading digits" therefore refers to leading *zero* (0) digits if the AFI value specifies that leading zero digits in the IDI are not significant; it refers to leading *one* (1) digits if the AFI value specifies that leading zero digits in the IDI *are* significant.

Note – The encodings defined in this section require that the IDI be padded to its maximum length, as described above, even when the value of the AFI specifies a decimal DSP syntax and the DSP is null.

A.8.3.1 *Preferred binary encoding*

The preferred binary encoding is generated by:

a) using two semi-octets to represent the two digits of the AFI, yielding a value for each semi-octet in the range 0000-1001;

b) padding the IDI with leading digits if necessary to obtain the maximum IDI length (specified for each IDI format in A.8.2.1.2), then using a semi-octet to represent the value of each decimal digit (including leading pad digits, if present), yielding a value in the range 0000-1001; and, if the DSP syntax is not decimal digits, using the semi-octet value 1111 as a pad after the final semi-octet (if necessary) to obtain an integral number of octets;

c) representing a decimal syntax DSP by using a semi-octet to represent the value of each decimal digit, yielding a value in the range 0000-1001 for each digit, and using the semi-octet value 1111 as a pad after the final semi-octet (if necessary) to obtain an integral number of octets;

d) representing a binary syntax DSP directly as binary octets;

e) when the IDI format is "Local", representing an ISO 646 character syntax DSP by converting each character to a number in the range 32-127 using the ISO 646 encoding, with zero parity and the parity bit in the most significant position, reducing the value by 32, giving a number in the range 0-95, encoding this result as a pair of decimal digits, and using a semi-octet to represent the value of each decimal digit, yielding a value in the range 0000-1001 for each digit, and

f) when the IDI format is "Local", representing a National Character syntax DSP by converting each national character to either one or two octets according to the rules specified by the authority responsible for allocating NSAP addresses including National Character DSP syntaxes.

A.8.3.2 *Preferred decimal encoding*

The preferred decimal encoding is generated by:

a) representing the two digits of the AFI directly as two decimal digits;

b) padding the IDI with leading digits if necessary to obtain the maximum IDI length (specified for each IDI format in A.8.2.1.2), representing the result directly as decimal digits;

c) representing a decimal syntax DSP directly as decimal digits;

d) representing a binary syntax DSP as follows: taking the octets in pairs, convert each octet of the pair into a number in the range 0-255; this generates six decimal digits, *abcdef*, of which digits *a* and *d* may take on only the values 0, 1, or 2. The pair of octets is represented by the sequence of five digits *gbcef*, in which the value of the digit *g* is given in Table A-4/X.213;

TABLE A-4/X.213

Values of «g»

d \ a	0	1	2
0	0	1	2
1	3	4	5
2	6	7	8

If the original binary field contained an odd number of octets, the final octet is converted to a number in the range 0-255 and represented as three decimal digits (000-255);

e) when the IDI format is "Local",, representing an ISO 646 character syntax DSP by converting each character to a number in the range 32-127 using the ISO 646 encoding, with zero parity and the parity bit in the most significant position, reducing the value by 32, giving a number in the range 0-95, encoding this result as a pair of decimal digits; and

f) when the IDI format is "Local", representing a National Character syntax DSP by converting each national character to either one or two octets according to the rules specified by the authority responsible for allocating NSAP addresses including National Character DSP syntaxes, and applying the technique described in A.8.3.2 d) above.

A.8.4 *Maximum Network address length*

The maximum length of the NSAP address for each of the combinations of IDI format and DSP abstract syntax is given in Table A-5/X.213 for both the preferred decimal encoding and the preferred binary encoding.

For this table it is clear that:

a) the maximum length of an NSAP address in its preferred binary encoding is *20 octets;* and

b) the maximum length of an NSAP address in its preferred decimal encoding is *40 digits.*

A Network layer protocol which is capable of conveying a string of variable length with a maximum length of either 20 binary octets or 40 decimal digits is capable of encoding the full semantic content of any Network address.

A.9 *Character based DSP allocation*

A network addressing authority may choose to allocate NSAP addresses with the DSP in a National Character syntax. In such cases, the allocating authority must define and publish the mapping of the National Character syntax to either the preferred binary encoding A.8.3.1) or the preferred decimal encoding A.8.3.2).

Note – It is recommended that this mappping be done by reference to the *ISO Register of Character Sets,* which is maintained by the European Computer Manufacturers Association (ECMA) acting as a registration authority according to ISO 2375.

TABLE A-5/X.213

Maximum NSAP address lengths

IDI format	DSP syntax	Binary DSP encoding (octets)	Decimal DSP encoding (digits)
X.121	Decimal	20	40
	Binary	17	39
ISO DCC	Decimal	20	40
	Binary	17	40
F.69	Decimal	20	40
	Binary	17	40
E.163	Decimal	20	40
	Binary	17	39
E.164	Decimal	20	40
	Binary	18	40
ISO 6523-ICD	Decimal	20	40
	Binary	16	39
Local	Decimal	20	40
	Binairy	16	40
	Character	20	40
	National character	15	37

Note – The National Character values assume a national character representation of one character as two binary octects.

In the case in which the authority defines and publishes the mapping of the national character set to a binary abstract syntax, the result must be representable in either one or two octets per national character. In this case, the resulting DSP is considered to be based on the binary abstract syntax. AFI values from Table A-2/X.213 and the mapping to preferred binary and decimal encodings are based on the binary abstract syntax.

In the case in which the authority defines and publishes the mapping of the national character set to a decimal abstract syntax, the result must be representable in up to five decimal digits per national character. In this case, the resulting DSP is considered to be based on the decimal abstract syntax. AFI values from Table A-2/X.213 and the mapping to preferred binary and decimal encodings are based on the decimal abstract syntax.

Note – The ability to base DSP allocation on national character sets allows DSP allocation based on international character set standards such as ISO 646, and also allows DSP allocation based on specific nationally recognized character sets. This may simplify address assignment in some cases, and may facilitate representation of NSAP addreses in humanly-readable form. Nevertheless, NSAP addresses should not be confused with Application layer entity titles. NSAP addresses are not intended to provide the same degree of human-readable, user-friendly naming and addressing capabilities as may be expected in Application layer entity titles.

A.10 *Reference publication formats*

Reference publication formats are defined to allow unambiguous representation of NSAP addresses in both written and oral communication.

A.10.1 *Decimal reference publication format*

The decimal reference publication format (DRPF) consists of a string of up to 40 decimal digits. The DRPF is the written inscription of the preferred decimal encoding defined in A.8.3.2.

A.10.2 *Hexadecimal reference publication format*

The hexadecimal reference publication format (HRPF) consists of the symbol "/" (solidus) followed by a string of up to 40 hexadecimal digits, in which the value of each binary octet in the preferred binary encoding defined in A.8.3.1 is represented as two hexadecimal digits.

A.11 *Network entity titles*

In order to perform routing functions and to distribute Network layer management information concerning routing among Network entities, it is necessary to be able to unambiguously identify Network entities in end systems and intermediate systems. Recommendation X.200 provides a definition of the concept of an (N)-entity title, which may be used to permanently and unambiguously identify a Network entity in an end system or intermediate system.

Any authority responsible for allocating addresses to NSAPs may choose also to allocate Network entity titles, following the same procedures and rules that it observes in the allocation of NSAP addresses. NSAP addresses and Network entity titles are syntactically indistinguishable; any value that the authority is permitted to allocate as an NSAP address may be allocated as a Network entity title.

APPENDIX I

(to Recommendation X.213)

Rationales for the material in Annex A

This appendix contains tutorial and explanatory material for Annex A.

I.1 *IDI formats* (A.8.2.1.2)

The rationale for the use of the specific IDI formats identified in A.8.2.1.2 is to allow the allocation and assignment of NSAP addresses to be based on existing, well-established network numbering plans and organization-identification standards.

The CCITT numbering plans are included so as to allow for the designation of the organization to which a number is assigned as an authority for the assignment of NSAP addresses. If the organization identified by a particular number from one of these plans chooses not to define any further subaddressing beyond that number, then the number itself constitutes an NSAP address when it is used in the OSI environment. This flexibility allows numbers allocated from the four CCITT numbering plans identified in A.8.2.1.2 to be used directly as NSAP addresses, with the addition of nothing more than the initial AFI digits that identify the plan.

The ISO DCC format is included so as to allow for the designation, where permitted by national regulations, of the organization that represents a crountry in ISO (or an appropriately sponsored organization) as an authority for the assignment of geographically-based NSAP addresses. The way in which addresses are allocated and assigned in the ISO DCC format is determined by the designated organization, which might, for example, be the national standards body that represents a country in ISO.

The ISO 6523-ICD format is included so as to allow for the designation, where permitted by national regulations, of an organization that may or may not be tied to a particular country as an authority for the assignment of NSAP addresses according to the hierarchy appropriate for the organization (which may not be based on geographical or national boundaries). The way in which addresses are allocated and assigned in the ISO 6523-ICD format is determined by the designated organization, which might, for example, be the United Nations World Healthe Organization. The ISO 6523-ICD format permits an organization already possessed of an ICD, for the purposes specified in ISO 6523, to use that ICD for the *additional* purpose of allocating Network addresses. This additional purpose is unrelated to the role of the ICD as the identifier of an Organization Code (OC) assignment scheme, which is the purpose for which ICDs are assigned. This does not change the criteria established in ISO 6523 for granting a request for an ICD allocation. Furthermore *only* the ICD is used in Network addresses that follow the ISO 6523-ICD IDI format. No part of any Network address corresponds to the OC defined in ISO 6523, and the OC is therefore not relevant to or affected by this.

The local format is included so as to allow for the coexistence of proprietary or other non-standard network addressing schemes with the standard OSI network addressing scheme. Use of the Local format for these non-standard addresses ensures that they cannot be confused with standard OSI network addresses. This capability will be useful in the evolution of existing networks to OSI, and for the accommodation of non-OSI addressing schemes that may be used in proprietary network architectures or for testing and other interim purposes. It should be emphasized that the Local format is not intended to give non-OSI schemes a permanent place in OSI, but rather to permit the OSI network addressing scheme to be used wherever possible without risk or conflict with other schemes (which can be encapsulated safely under the Local format).

I.2 *Reservation of AFI values 00-09* (Table A-2/X.213)

The reservation of AFI values beginning with the digit 0 is intended to allow for the use of an initial 0 handle special cases, such as:

a) as an escape to some other addressing scheme;

b) as a technique for the optimization of NSAP address encoding in Network layer protocols, when different parts of NSAP address semantics are encoded in different fields of the protocol header;

c) as a way to indicate, in a protocol header, that a field that ordinarily contains a full NSAP address in fact contains something less than a full address (for example, a shorthand form that omits specification of the higher-order addressing domain(s), which might be used for communication within a particular subdomain environment).

There may be other cases in which the use of an initial zero digit is found to be useful. Annex A merely reserves the AFI values 00-09, and does not specify how they might be used; all such uses are outside the scope of Annex A.

I.3 *Derivation of the preferred encodings* (A.8.3)

In describing the two preferred encodings of the NSAP address, A.8.3.1 and A.8.3.2 introduce two types of padding: padding with non-significant leading zero or one digits at the beginning of an IDI, and padding with a semi-octet with the value 1111 at the end of the binary encoding of an IDI with an odd number of decimal digits.

The first type of padding is necessary because in some formats the IDI consists of a variable number of digits. Since there is no explicit syntactic marker between the IDI and the DSP, the only way to find the boundary between them is to know how long the IDI is. The AFI, which identifies the IDI format, specifies only the *maximum* length of the IDI in that format. Rather than introduce either a specifc syntactic marker or a new field containing the length of the IDI (either of which would complicate the encoding and parsing of NSAP addresses), Annex A specifies that for encoding purposes the IDI must first be padded out to its maximum length. Note that this does *not* apply to the DSP; only to the IDI.

The second type of padding is necessary to ensure that a binary encoding of the IDI consists of an integral number of binary octets.

APPENDIX II

(to Recommendation X.213)

Differences between Recommendation X.213 and ISO 8348

The following differences between this Recommendation and ISO 8348 should be noted.

II.1 The following note, which is contained in § 12.2.7.2, does not appear in ISO 8348.

"*Note* – The implementation of the transit delay negotiation requires urgent further study in order to have a harmonized realization in different types of subnetworks. Special attention is required as regards routing and charging consequences."

II.2 The following note, which is contained in § 12.2.8, does not appear in ISO 8348.

"*Note* – The objective is to make this parameter a mandatory parameter to be supported by all subnetworks in the future. However, a number of existing subnetworks cannot support it now. During the interim period, while these subnetworks exist and are not modified to provide this parameter, it is considered as a provider-option. No negotiation mechanism is needed in the Network Service. Limiting, in some subnetworks, the length of NS-user-data to be provided to a value lower than 128 octets (e.g. 16 to 32 octets) for an interim period would imply fewer changes to existing interfaces and signalling systems and would simplify the introduction of such a service in existing subnetworks."

In addition, in Table 6/X.213, the NS-user-data parameters are marked as "conditional", whereas these parameters are not marked as "conditional" in ISO 8348.

II.3 The following note, which is contained in § 13.2.3, does not apper in ISO 8348.

"*Note* – The objective is to make this parameter a mandatory parameter to be supported by all subnetworks in the future. However, a number of existing subnetworks cannot support it now. During the interim period, while these subnetworks exist and are not modified to provide this parameter, it is considered as a provider-option. No negotiation mechanism is needed in the Network Service".

In addition, in Table 13/X.213, the NS-user-data parameters are marked as "conditional", whereas these parameters are not marked as "conditional" in ISO 8348.

II.4 The material contained in Annex A and Appendix I to this Recommendation is contained in ISO 8348/Add.2 and its Appendix.

Recommendation X.214

TRANSPORT SERVICE DEFINITION 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.224 specifies the connection-oriented Transport Protocol Specification for Open Systems Interconnection for CCITT Applications;

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

[1)] Recommendation X.214 and ISO 8072 [Information Processing Systems – Open Systems Interconnection – Transport Service Definition] were developed in close collaboration and are technically aligned.

(d) that Recommendation X.225 specifies the connection-oriented Session Protocol Specification for Open Systems Interconnection for CCITT applications;

(e) that Recommendation X.210 specifies the OSI Layer Service Definition Conventions for describing the services of the layers of the OSI Reference Model,

unanimously declares

(1) that the scope, field of application, and related definitions and abbreviations of the Open Systems Interconnection Transport Service Definition are defined in §§ 1 to 4;

(2) that the conventions for describing the Transport Service are given in § 5;

(3) that the overview, general characteristics, and features of the Transport Service and the Classes of Transport Service are described in §§ 6, 7 and 8;

(4) that the model of the Transport Service is described in § 9;

(5) that the quality of Transport Service is defined in § 10;

(6) that the Transport Service primitives and their related parameters are desired in §§ 11, 12, 13 and 14;

(7) that the formal description of the Transport Service is contained in § 15.

CONTENTS

0 Introduction

This Recommendation is one of a set of Recommendations produced to facilitate the interconnection of computer systems. It is related to other Recommendations in the set as defined by the Reference Model of Open Systems Interconnection (OSI). The OSI Reference Model (Reference 1) subdivides the area of standardization for interconnection into a series of layers of specification, each of manageable size.

This Recommendation defines the Service provided by the Transport Layer to the Session Layer at the boundary between the Transport and Session Layers of the Reference Model. It provides for the designers of Session Protocols a definition of the Transport Service existing to support the Session Protocol and for designers of Transport Protocols a definition of the services to be made available through the action of the Transport Protocol over the underlying service. This relationship is illustrated in Figure 1/X.214.

Throughout the set of OSI Recommendations, the term "Service" refers to the abstract capability provided by one layer of the OSI Reference Model to the layer above it. Thus, the Transport Service defined in this Recommendation 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 Recommendations from its use elsewhere to describe the provision of a service by an organisation (such as the provision of a service as defined in other CCITT Recommendations by an Administration).

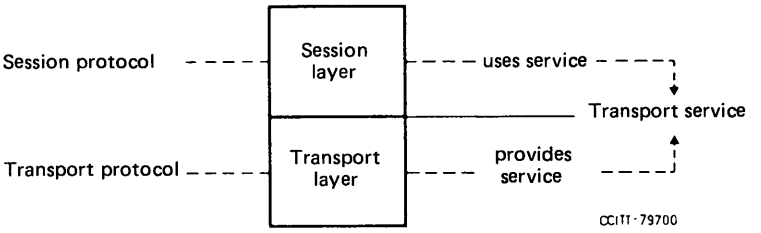


FIGURE 1/X.214

Relationship of this Recommendation to other OSI Recommendations

1 Scope and field of application

This Recommendation defines in an abstract way the externally visible service provided by the OSI Transport Layer in terms of:

a) the primitive actions and events of the service;

b) the parameter data associated with each primitive action and event;

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

The service defined in this Recommendation is that which is provided by all OSI Transport Protocols (in conjunction with the Network Service) and which may be used by any OSI Session Protocol.

This Recommendation does not specify individual implementations or products, nor does it constrain the implementation of entities and interfaces within a system. Conformance of equipment to this Recommendation is achieved by conformance to the protocols specified to fulfill the Transport Service defined in this Recommendation.

2 References

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

[2] Recommendation X.224 – Transport Protocol Specification for Open Systems Interconnection for CCITT Applications (see also ISO 8073).

[3] Recommendation X.225 – Session Protocol Specification for Open Systems Interconnection for CCITT Applications (see also ISO 8327).

[4] Recommendation X.210 – OSI Layer Service Definition Conventions (see also ISO TR 8509).

[5] Recommendation T.70 – Network – Independent Basic Transport Service for Teletex.

3 Definitions

3.1 *Reference Model definitions*

This Recommendation is based on the concepts developed in the OSI Reference Model (Reference 1), and makes use of the following terms defined in that Recommendation:

a) expedited transport-service-data-unit;

b) transport-connection;

c) transport-connection endpoint;

d) Transport Layer;

e) Transport Service;

f) transport-service-access-point;

g) transport-service-access-point address;
h) transport-service-data-unit;
i) Network Layer;
j) Network Service;
k) network-connection;
l) interface flow control.

3.2 *Service Definition Convention*

This Recommendation also makes use of the following terms defined in Reference 4, OSI Layer Service Definition Conventions as they apply to the Transport Layer:

a) service-user;
b) service-provider;
c) primitive;
d) request;
e) indication;
f) response;
g) confirm.

3.3 *Transport Service definitions*

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

3.3.1 **Calling TS user**

A Transport Service user that initiates a transport connection establishment request.

3.3.2 **Called TS user**

A Transport Service user with whom a calling TS user wishes to establish a transport-connection.

Note – Calling TS users and called TS users are defined with respect to a single connection. A Transport Service user can be both a calling and a called TS user simultaneously.

3.3.3 *Sending TS user*

A Transport Service user that acts as a source of data during the data transfer phase of a transport-connection.

3.3.4 *Receiving a TS user*

A Transport Service user that acts as a sink of data during the data transfer phase of a transport-connection.

Note – A Transport Service user can be both a sending and a receiving TS user simultaneously.

4 Abbreviations

TS: Transport Service
TC: Transport-connection
TSAP: Transport-service-access-point
TSDU: Transport-service-data-unit
QOS: Quality of Service

5 Conventions

5.1 *General conventions*

This Recommendation uses the descriptive conventions given in Reference 4, OSI Layer Service Definition Conventions.

5.2 *Parameters*

The available parameters for each group of primitives are set out in tables in §§ 12 to 14. Each "X" in the tables indicates that the primitive labelling the column in which it falls may carry the parameter labelling the row in which it falls.

Some entries are further qualified by items in brackets. These may be:

a) indications that the parameter is optional in some way:

(U) indicates that the inclusion of the parameter is a choice made by the user;

b) a parameter specific constraints:

(=) indicating that the value supplied in an indication or confirm primitive is always identical to that supplied in the previous request or response primitive issued at the peer service access point.

6 Overview and general characteristics

The Transport Service provides transparent transfer of data between TS users. It relieves these TS users from any concern about the detailed way in which supporting communications media are utilized to achieve this transfer.

The Transport Service provides for the following:

a) Quality of Service selection:

The Transport Layer is required to optimize the use of available communications resources to provide the Quality of Service required by communicating TS users at minimum cost. Quality of Service is specified through the selection of values for Quality of Service parameters representing characteristics such as throughput, transit delay, residual error rate and failure probability.

b) Independence of underlying communications resources:

The Transport Service hides from TS users the difference in the Quality of Service provided by the Network Service. This difference in Quality of Service arises from the use of a variety of communications media by the Network Layer to provide the Network Service.

c) End-to-end significance:

The Transport Service provides for the transfer of data between two TS users in end systems.

d) Transparency of transferred information:

The Transport Service provides for the transparent transfer of octet-aligned TS user-data and/or control information. It does neither restrict the content, format, or coding of the information, nor does it ever need to interpret its structure or meaning.

e) TS user addressing:

The Transport Service utilizes a system of addressing which is mapped into the addressing scheme of the supporting Network Service. Transport-addresses can be used by TS users to refer unambiguously to TSAPs.

7 Features of the Transport Service

The Transport Service offers the following features to a TS user:

a) The means to establish a TC with another TS user for the purpose of exchanging TSDUs. More than one TC may exist between the same pair of TS users.

b) Associated with each TC at its time of establishment, the opportunity to request, negotiate, and have agreed by the TS provider a certain Quality of Service as specified by means of Quality of Service parameters.

c) The means of transferring TSDUs on a TC. The transfer of TSDUs which consist of an integral number of octets is transparent, in that the boundaries of TSDUs and the contests of TSDUs are preserved unchanged by the TS provider and there are no constraints on the TSDU content imposed by the TS provider.

d) The means by which the receiving TS user may control the rate at which the sending TS user may send octets of data.

e) The means of transferring separate expedited TSDUs when agreed to by both TS users. Expedited TSDUs transfer is subject to a different flow control from normal data across the TSAP.

f) The unconditional and therefore possible destructive release of a TC.

8 Classes of Transport Service

No distinct classes of Transport Service are defined.

9 Model of the Transport Service

9.1 *Model of the Transport Service*

This Recommendation uses the abstract model for a layer service defined in Reference 4 Service Conventions. The model defines the interactions between the TS users and the TS provider which take place at the two TSAPs. Information is passed between a TS user and the TS provider by service primitives, which may convey parameters.

The primitives are abstract representations of TSAP interactions. They are solely descriptive and do not represent a specification for implementation.

9.2 *Model of a Transport Connection*

The operation of a TC is modelled in an abstract way by a pair of queues linking the two TSAPs. There is one queue for each direction of information flow (see Figure 2/X.214). Each TC is modelled by a separate pair of queues.

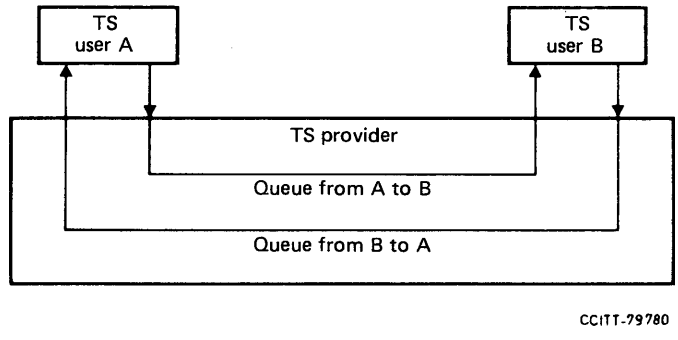


FIGURE 2/X.214

Abstract Model of a Transport Connection

The queue model is used to introduce the flow control feature. The ability of a TS user to add objects to a queue will be determined by the behaviour of the TS user removing objects from that queue and the state of the queue. Objects are entered and removed from the queue as a result of interactions at the two TSAPs.

The pair of queues is considered to be available for each potential TC.

The objects which may be placed in a queue by a TS user (see §§ 12, 13 and 14) are:

a) connect objects (each representing all parameters contained in a T-CONNECT request or T-CONNECT response primitive);

b) octets of normal data;

c) indications of end-of-TSDU (completion of a T-Data primitive);

d) expedited TSDUs (representing all parameters of a T-EXPEDITED primitive);

e) disconnect objects (each representing all parameters contained in a T-DISCONNECT primitive).

Note 1 – Normal and expedited TSDU transfer will result in different objects being entered into the queue.

Note 2 – The description of flow control requires a less abstract description than that used for describing sequences of primitives in §§ 11-14. Each TSDU associated with a T-DATA primitive is here subdivided conceptually into a sequence of octets of data followed by an end-of-TSDU indication. The T-Data request primitive occurs when the end-of-TSDU indication is entered into the queue. The T-DATA indication primitive occurs when the end-of-TSDU indication is removed from the queue. This does not imply any particular subdivision in any real interface.

The only objects which can be placed in a queue by the TS provider are disconnect objects (representing T-DISCONNECT primitives and their parameters).

TS user A, who initiates connection establishment by entering a connect object (representing a T-CONNECT request primitive) into the queue from A to B, is not allowed to enter any other object into this queue until after the connect object representing the T-CONNECT confirm has been removed. In the queue from TS user B to TS user A, objects other than a disconnect object can be entered by TS user B only after TS user B has entered a connect object corresponding to a T-CONNECT response. The insertion of a disconnect object represents the initiation of the release procedure. The release procedure may be initiated at the times permitted in § 14 and in the manner described in § 11.2. The release procedure may be destructive with respect to other objects in the two queues.

A queue relates an ordered set of distinct objects in the following ways:

a) Queues are empty before a connect object has been added and can be returned to this state, with loss of their contents, by the TS provider under the circumstances as described in h) below.

b) Objects are added to the queue, subject of control by the TS provider.

c) Objects are normally removed from the queue, subject to control by the receiving TS user.

d) Objects are normally removed in the same order that they were added [but see g) and h) below].

e) A queue has a limited capacity, but this capacity is not necessarily either fixed or determinable.

f) The management of the queue capacity shall be such that normal data and end-of-TSDU indications cannot be added to the queue when its addition would prevent addition of an expedited TSDU or disconnect object.

In addition the TS provider may manipulate pairs of adjacent objects in the queue to allow:

g) Reordering

The order of any pair of objects may be reversed if, and only if, the following object is of a type defined to take precedence over the preceding object. Expedited TSDUs take precedence over octets of normal data and end-of-TSDU indications (see Table 1/X.214).

TABLE 1/X.214

Precedence Table

has precedence over queue object y / the queue object x	Connect object	Octets of normal data	End-of-TSDU indication	Expedited TSDU	Disconnect object
Connect object	–	No	–	No	Yes [see h)]
Octets of normal data	–	No	No	Yes [see g)]	Yes [see h)]
End-of-TDSU indication	–	No	No	Yes [see g)]	Yes [see h)]
Expedited TSDU	–	No	No	No	Yes [see h)]
Disconnect object	–	–	–	–	Yes [see h)]

–: not applicable.
No: no precedence exists.
Yes: precedence exists.

h) Deletion

Disconnect objects take precedence over any other object. Any object other than a disconnect object may be deleted by the TS provider if, and only if, the following one is a disconnect object (see Table 1/X.214).

If a connect object associated with a T-CONNECT request primitive is deleted in this manner, the disconnect object is also deleted. If a connect object associated with a T-CONNECT response positive is deleted, the disconnect object is not deleted.

Whether the TS provider performs actions of types g) and h) or not, will depend on the behaviour of the TS users and on the agreed Quality of Service. In general, if the objects are not removed from the queue due to flow control expressed by the receiving TS user, the TS provider shall, after some unspecified period of time, perform all permitted actions of types g) and h).

Note 1 – The internal mechanisms which support the operation of a queue are not visible in the Transport Service. A queue is one particular way of expressing the mutual interaction between primitives at different TSAPs. There may also be, for example:

a) constraints on the local ability to invoke primitives;

b) service procedures defining particular sequencing constraints on some primitives.

Note 2 – A TC endpoint identification mechanism must be provided locally if the TS user and the TS provider need to distinguish between several TCs at a TSAP. All primitives must then make use of this identification mechanism to identify the TC to which they apply. This implicit identification is not shown as a parameter of the TS primitives, and must not be confused with the address parameters of the T-CONNECT primitives.

10 Quality of Transport Service

The term Quality of Service (QOS) refers to certain characteristics of a TC as observed between the endpoints.

QOS is described in terms of QOS parameters.

These parameters give TS users a method of specifying their needs, and give the TS provider a basis for protocol selection.

The QOS is normally negotiated between the TS users and the TS provider on a per TC basis, using the T-CONNECT request, indication, response, and confirm TS primitives defined in § 11. The QOS requested by the calling TS user may be made poorer either by the TS provider following the T-CONNECT request, or by the called TS user, following the T-CONNECT indication. In applying this to some QOS parameters this may mean that:

a) a delay becomes longer,

b) a throughput becomes lower,

c) the error rate becomes higher,

d) the priority becomes lower,

e) the failure probability becomes higher.

However the TC protection parameter remains unchanged by the TS provider.

The so negotiated QOS values then to apply throughout the lifetime of the TC.

Note – Users of the Transport Service should be aware that there is no guarantee that the originally negotiated QOS will be maintained throughout the Transport Connection lifetime, and that changes in QOS are not explicitly signalled by the Transport Service provider.

The view of QOS at each end of an established TC is always the same.

This section does not specify particular values, or classes of values, for the QOS parameters. Possible choices and default values for each parameter will normally be specified at the time of initial TS provider installation. The values for any or all parameters may be fixed for a given TS provider, in which case QOS negotiation on a per TC basis is not required. When a QOS value is specified; the TS user may also indicate whether the request is an absolute requirement or whether a degraded value is acceptable.

The QOS parameters include parameters which express TS performance and parameters which express other TS characteristics.

The QOS parameters specified in this clause are defined below. A classification of the performance QOS parameters is shown in Table 2/X.214.

10.1 *TC establishment delay*

TC establishment delay is the maximum acceptable delay between a T-CONNECT request and the corresponding T-CONNECT confirm primitive.

Note – This delay includes TS user dependent components.

10.2 *TC establishment failure probability*

TC establishment failure probability is the ratio of total TC establishment failures to total TC establishment attempts in a measurement sample.

A TC establishment failure is defined to occur when a requested TC is not established within the specified maximum acceptable TC establishment delay as a result of misconnection, TC refusal, or excessive delay on the part of the TS provider. TC establishment attempts which fail as a result of error, TC refusal, or excessive delay on the part of a TS user are excluded in calculating the TC establishment failure probability.

TABLE 2/X.214

Classification of Performance QOS Parameters

Phase	Performance criterion	
	Speed	Accuracy/Reliability
TC establishment	TC establishment delay	TC establishment failure probability (misconnection/TC refusal)
Data transfer	Throughput Transit delay	Residual error rate (corruption, duplication/loss) Resilience of the TC Transfer failure probability
TC release	TC release delay	TC release failure probability

10.3 *Throughput*

Throughput is defined, for each direction of transfer, in terms of a sequence of at least two successfully transferred TSDUs. Given such a sequence of n TSDUs, where n is greater than or equal to two, the throughput is defined to be the smaller of:

a) the number of TS user data octets contained in the last $n - 1$ TSDUs divided by the time between the first and last T-DATA requests in the sequence; and

b) the number of TS user data octets contained in the last $n - 1$ TSDUs divided by the time between the first and last T-DATA indicators in the sequence.

Successful transfer of the octets in a transmitted TSDU is defined to occur when the octets are delivered to the intended receiving TS user without error, in the proper sequence, prior to release of the TC by the receiving TS user.

Throughput is only meaningful for a sequence of complete TSDUs and each specification is based on a previously stated average TSDU size.

Throughput is specified separately for each direction of transfer on a TC. In each direction, a specification of throughput will consist of a *maximum throughput* and an *average throughput* value. The *maximum throughput* value represents the maximum rate at which the TS provider can continuously accept and deliver TSDUs, in the absence of sending TS user input delays or flow control applied by the receiving TS user. Thus, the sequence of TSDUs in the calculation above are defined to be presented continuously at the maximum rate. The *average throughput* value represents the expected transfer rate on a TC including the effects of expected user-attributable delays (e.g. non-continuous TSDU input, receiving TS user flow control). Thus, the sequence of TSDUs in the calculation above are defined to be presented at a rate which includes components representing *average* user delays.

It is possible for either the input or the output of a sequence of TSDUs to be excessively delayed by the TS users. Such occurrences are excluded in calculating *average throughput* values.

For each direction of transfer, and for each of the *maximum throughput* and *average throughput* specifications, the throughput QOS for a particular TC will be negotiated among the TS users and the TS provider (see § 12.2.6).

10.4 *Transit delay*

Transit delay is the elapsed time between a T-DATA request and the corresponding T-DATA indication. Elapsed time values are calculated only on TSDUs that are successfully transferred.

Successful transfer of a TSDU is defined to occur when the TSDU is transferred from the sending TS user to the intended receiving TS user without error, in the proper sequence, prior to release of the TC by the receiving TS user.

Transit delay is specified independently for each direction of transfer. In general, each transit delay specification will define both the average value and the maximum value expected for a TC. Each specification will be based on a previously stated average TSDU size.

The transit delay for an individual TSDU may be greatly increased if the receiving TS user exercises interface flow control. Such occurrences are excluded in calculating both average and maximum transit delay values.

10.5 *Residual error rate*

Residual error rate is the ratio of total incorrect, lost and duplicate TSDUs to total TSDUs transferred across the TS boundary during a measurement period. The relationship among these quantities is defined for a particular TS user pair, as shown in Figure 3/X.214.

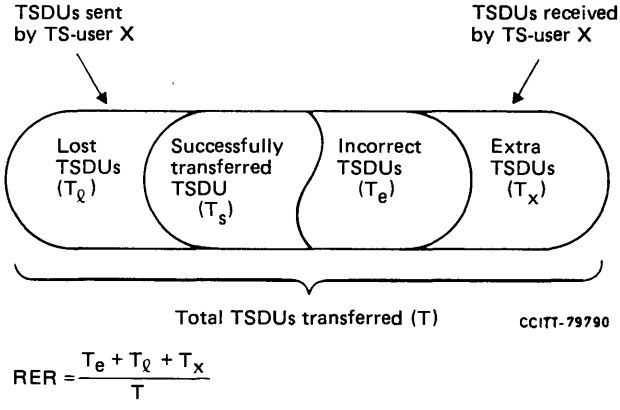


FIGURE 3/X.214

Components of Residual Error Rate

10.6 *Transfer failure probability*

Transfer failure probability is the ratio of total transfer failures to total transfer samples observed during a performance measurement.

A transfer sample is a discrete observation of TS provider performance in transferring TSDUs between a specified sending and receiving TS user. A transfer sample begins on input of a selected TSDU at the sending TS user boundary, and continues until the outcome of a given number of TSDU transfer attempts has been determined. A transfer sample will normally correspond to the duration of an individual TC.

A transfer failure is a transfer sample in which the observed performance is worse than a specified minimum acceptable level. Transfer failures are identified by comparing the measured values for three supported performance parameters with specified transfer failure thresholds. The three supported performance parameters are throughput, transit delay and residual error rate.

In systems where Transport Service QOS is reliably monitored by the TS provider, transfer failure probability can be estimated by the probability of a TS provider initiated release during a transfer sample.

10.7 *TC release delay*

TC release delay is the maximum acceptable delay between a TS user initiated T-DISCONNECT request and the successful release of the TC at the peer TS user. TC release delay is normally specified independently for each TS user. TC release delay does not apply in cases where release is initiated by the TS provider.

Issuance of a T-DISCONNECT request by either TS user starts the counting of TC release delay for the other user. Successful release is signalled to the TS user not initiating the T-DISCONNECT request by a T-DISCONNECT indication.

10.8 *TC release failure probability*

TC release failure probability is the ratio of total TC release requests resulting in release failure to total release requests included in a measurement sample. TC release failure probability is normally specified independently for each TS user.

A release failure is defined to occur, for a particular TS user, if that user does not receive a T-DISCONNECT indication within the specified maximum TC release delay of the TS user issuing the T-DISCONNECT request (given that the former TS user had not itself issued a T-DISCONNECT request).

10.9 *TC protection*

TC protection is the extent to which a TS provider attempts to prevent unauthorized monitoring or manipulation of TS user originated information. TC protection is specified qualitatively by selecting one of four TC protection options:

a) no protection features;

b) protection against passive monitoring;

c) protection against modification, replay, addition or deletion;

d) both b) and c).

10.10 *TC priority*

The specification of TC priority is concerned with the relationship between TCs. This parameter specified that relative importance of a TC with respect to:

a) the order in which TCs are to have their QOS degraded, if necessary, and

b) the order in which TCs are to be broken to recover resources, if necessary.

This parameter only has meaning in the context of some management entity or structure able to judge relative importance. The number of priority levels is limited.

10.11 *Resilience of the TC*

Probability of a TS provider initiated TC release (i.e. issuance of a T-DISCONNECT indication with no prior T-DISCONNECT request) during a specified time interval (e.g.: 1 s).

11 Sequence of Transport Service primitives

This section defines the constraints on the sequences in which the TS primitives may occur. The constraints determine the order in which TS primitives occur, but do not fully specify when they may occur. Other constraints, such as flow control of data, will affect the ability of a TS user or TS provider to issue a TS primitive at any particular time.

§§ 12-14 describe the TS primitives which are associated with one of the three phases of a TC, establishment, data transfer, or release. A complete listing of TS primitives appears in Table 3/X.214.

11.1 *Relation of TS primitives at the two TC endpoints*

A TS primitive issued at one TC endpoint will, in general, have consequences at the other TC endpoint. The relations of TS primitives of each type to TS primitives at the other TC endpoint are defined in the appropriate subclause in §§ 12-14; all these relations are summarized in the Figure 4/X.214 below (see Reference 4, § 5 for the definition of time sequence diagrams). However, a T-DISCONNECT request or indication TS primitive may terminate any of the other sequences before completion.

11.2 *Sequence of TS primitives at one TC endpoint*

The possible overall allowed sequences of TS primitives at a TC endpoint are defined in the following state transition diagram (see Figure 5/X.214) and in an alternative tabular representation (see Table 4/X.214).

In Figure 5/X.214:

a) The idle state (1) reflects the absence of a TC. It is the initial and final state of any sequence and once it has been re-entered, the TC is released.

b) A TC release procedure can be initiated at any point during the TC establishment or data transfer phase.

c) Procedures other than the TC release procedure cannot be initiated within the establishment phase.

d) Any action to be taken on the occurrence of a non-allowed sequence of TS primitives is a local matter.

e) The use of a state transition diagram to describe the allowable sequences of TS primitives does not impose any requirement or constraint on the internal organization of any implementation of the Transport Service.

12 Transport Connection establishment phase

12.1 *Function*

The TC establishment TS primitives can be used to establish a TC, provided the TS users exist and are known to the TS provider.

Simultaneous T-CONNECT requests at the two TSAPs are handled independently by the TS provider.

Note – Simultaneous T-CONNECT requests typically result in a corresponding number of TCs.

TABLE 3/X.214

Transport Service Primitives

Phase	Service	Primitive	Parameters
TC establishment	TC establishment	T-CONNECT request	(Called address, calling address, expedited data option, quality of service TS user-data)
		T-CONNECT indication	(Called address, calling address, expedited data option, quality of service TS user-data)
		T-CONNECT response	(Quality of service, responding address, expedited data option, TS user-data)
		T-CONNECT confirm	(Quality of service, responding address, expedited data option, TS user-data)
Data transfer	Normal data transfer	T-DATA request	(TS user-data)
		T-DATA indication	(TS user-data)
	Expedited data transfer [a]	T-EXPEDITED-DATA request	(TS user-data)
		T-EXPEDITED-DATA indication	(TS user-data)
TC release	TC release	T-DISCONNECT request	(TS user-data)
		T-DISCONNECT indication	(Disconnect reason, TS user-data)

[a] User option: Provided only upon TS user request.

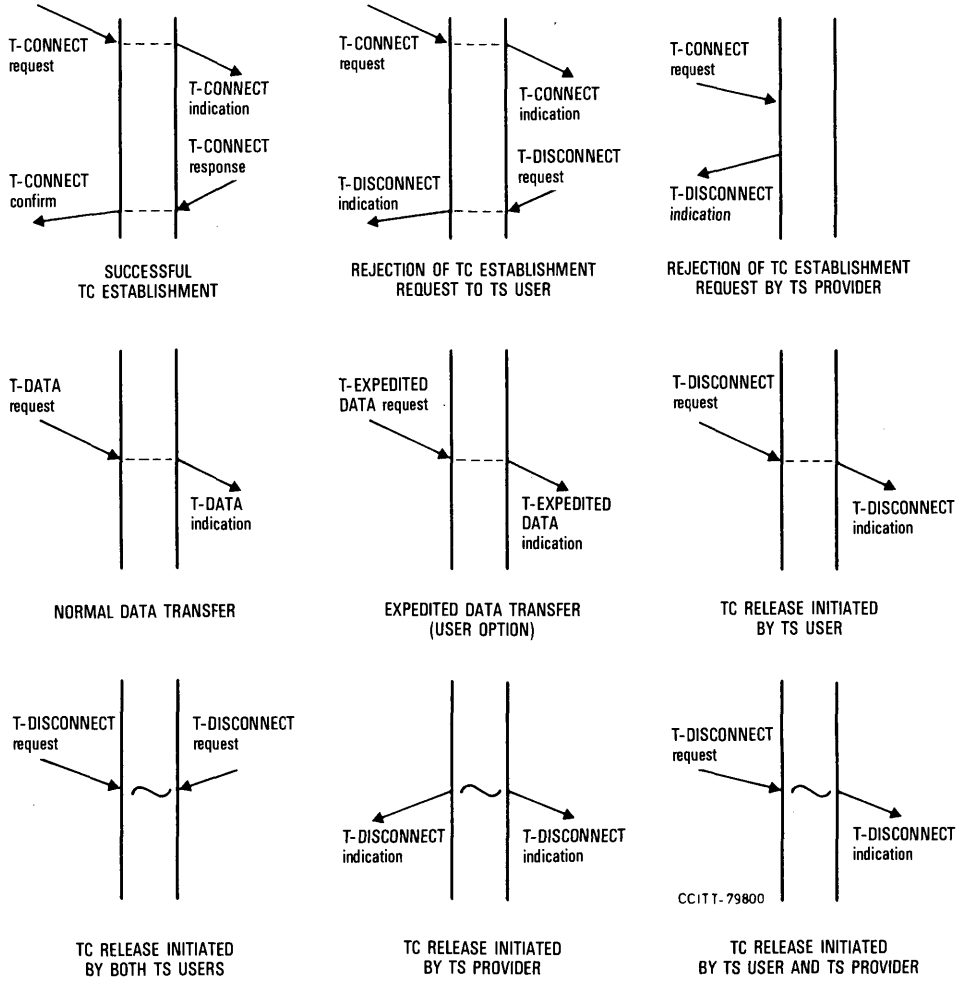


FIGURE 4/X.214

Summary of Transport Service Primitive Time Sequence Diagrams

TABLE 4/X.214

Sequences of TS Primitives at one End of a TC

may be followed by the TS primitive Y \ The TS primitive X	T-CONNECT request	T-CONNECT confirm	T-CONNECT indication	T-CONNECT response	T-DATA request	T-DATA indication	T-EXPEDITED-DATA request	T-EXPEDITED-DATA indication	T-DISCONNECT request	T-DISCONNECT indication
T-CONNECT request										
T-CONNECT confirm	+									
T-CONNECT indication										
T-CONNECT response			+							
T-DATA request		+		+	+	+	+	+		
T-DATA indication		+		+	+	+	+	+		
T-EXPEDITED-DATA request		+		+	+	+	+	+		
T-EXPEDITED-DATA indication		+		+	+	+	+	+		
T-DISCONNECT request	+	+	+	+	+	+	+	+		
T-DISCONNECT indication	+	+	+	+	+	+	+	+		

\+ : Possible
blank : Not possible.

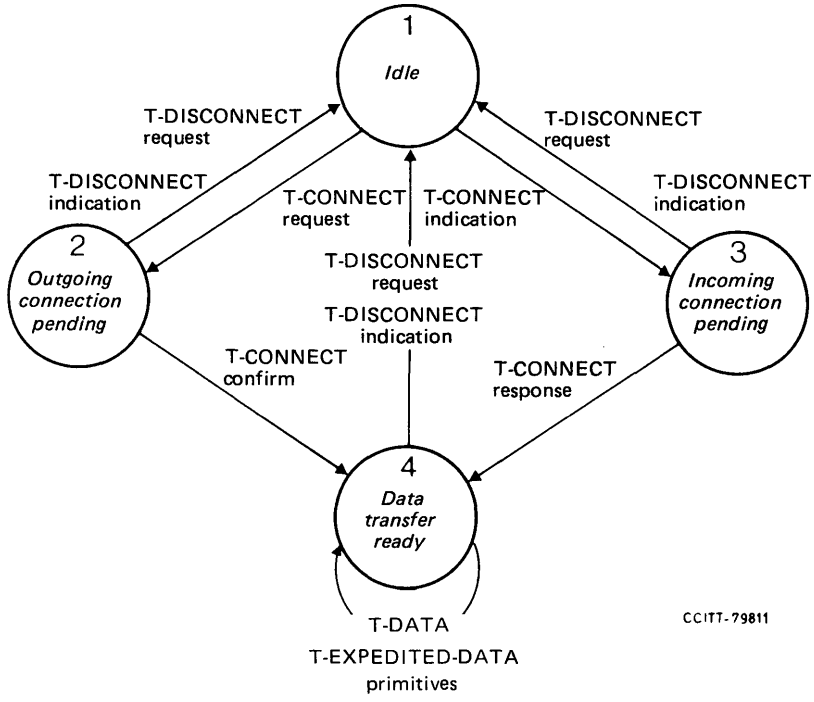


FIGURE 5/X.214

State Transition Diagram for Possible Allowed Sequence of TS Primitives at a TC Endpoint

12.2 *Types of TS primitives and parameters*

The following Table 5/X.214 indicates the types of TS primitives and the parameters needed for TC establishment.

12.2.1 *Addresses*

The parameters which take addresses as values (see §§ 12.2.2 to 12.2.4) all refer to TSAPs. These addresses are unique within the scope of TSAP addresses.

12.2.2 *Called Address*

The Called Address parameter conveys the address of the TSAP to which the TC is to be established.

12.2.3 *Calling Address*

The Calling Address parameter conveys the address of the TSAP from which the TC has been requested.

12.2.4 *Responding Address*

The Responding Address parameter conveys the address of the TSAP to which the TC has been established and is identical to the Called Address parameter.

Note – This parameter may be used in future definitions to return an address which is different from the Called Address, e.g., a specific address returned as the actual result of calling a generic address.

TABLE 5/X.214

TC Establishment Primitives and Parameters

Parameter \ TS-Primitive	T-CONNECT request	T-CONNECT indication	T-CONNECT response	T-CONNECT confirm
Called address	X	X		
Calling address	X	X(=)		
Responding address			X	X(=)
Expedited data option	X	X(=)	X	X(=)
Quality of service	X	X	X	X(=)
TS user-data	X(U)	X(=)	X(U)	X(=)

X: mandatory parameter.
(=): the value of that parameter is identical to the value of the corresponding parameter of the preceding TS primitive.
(U): use of this parameter is a TS user option.

12.2.5 *Expedited Data option*

The Expedited Data option parameter indicates whether the expedited data option is to be available on the TC. If this service is declared not available, it cannot be used on the TC. The value of the parameter is either "Expedited Data Service selected" or "Expedited Data Service not selected" (see § 12.4). The value of the various primitives are related such that:

a) in the T-CONNECT request primitive, either of the defined values may occur;

b) in the T-CONNECT indication primitive, the value is equal to the value in the T-CONNECT request primitive;

c) in the T-CONNECT response primitive, the value is either "Expedited Data Service not selected" or is equal to the value in the T-CONNECT indication primitive;

d) in the T-CONNECT confirm primitive, the value is equal to the value in the T-CONNECT response primitive.

12.2.6 *Quality of Service*

Quality of Service is a list of parameters (see § 10). For each parameter, the values in the various TS primitives are related so that:

a) in the T-CONNECT request primitive, any defined value is allowed,

b) in the T-CONNECT indication primitive, the QOS parameter value is equal to or poorer than the value in the T-CONNECT request primitive, except for the TC protection, which must have the same value as specified in the T-CONNECT request primitive,

c) in the T-CONNECT response primitive, the QOS parameter value indicated is equal to or poorer than the value in the T-CONNECT response primitive,

d) in the T-CONNECT confirm primitive, the QOS parameter value indicated is equal to the value in the T-CONNECT response primitive.

12.2.7 *TS User-Data*

This parameter allows the transfer of TS user-data between TS users without modification by the TS provider. The TS user-data parameter shall be an integral number of octets in length between 1 and 32 inclusive.

Note 1 – The called TS user may use the information conveyed to determine whether or not the TC should be accepted.

Note 2 – The QOS associated with TS user-data on the T-CONNECT primitive may be lower than that for TS user-data in the T-DATA primitive once the TC is established.

12.3 *Sequence of TS primitives*

The sequence of TS primitives in a successful TC establishment is defined by the following time sequence diagram (Figure 6/X.214):

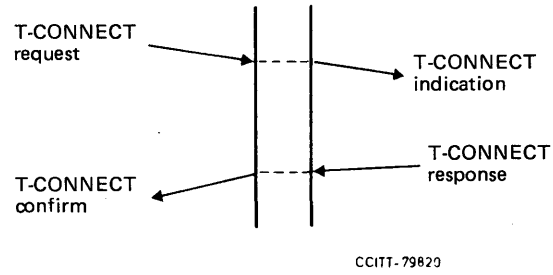


FIGURE 6/X.214

Sequence of Primitives in Successful TC Establishment

The TC establishment procedure may fail either due to the inability of the TS provider to establish a TC or due to the unwillingness of the called TS user to accept a T-CONNECT indication. These cases are described in §§ 14.4 and 14.5. The TC establishment procedure may also fail due to either of the TS users releasing the TC before the T-CONNECT confirm has been delivered to the calling TS user.

12.4 *Negotiation of expedited data Transfer Service*

The expedited TSDU transfer is only made available when specifically requested and agreed to by both TS users when the TC is established. When available this service is always bidirectional. The procedure for negotiating the use of the expedited TSDU transfer is as follows:

a) the calling TS user may request or not request the use of the expedited TSDU transfer feature;

b) if the calling TS user does not request the use of the expedited TSDU transfer feature, the called TS user is not allowed to request its use;

c) if the calling TS user does request the use of the expedited TSDU transfer feature, the called TS user may agree to the use of expedited TSDU transfer on the TC, in which case the TS provider is required to provide it. The called TS user may refuse the use of expedited TSDU transfer in which case the service will not be used on that TC.

13 Data transfer phase

13.1 *Data Transfer Service*

13.1.1 *Function*

The TS provider provides for an exchange of TSDUs, in both directions simultaneously. The TS provider preserves the integrity, the sequence and boundaries of the TSDUs.

Note – Designers of higher layer protocols should realize that the requested QOS applies to complete TSDUs, and that division of data into small TSDUs may have cost implications, because of the impact on cost optimization mechanisms operated by the TS provider.

13.1.2 *Types of TS primitives and parameters*

Table 6/X.214 indicates the types of TS primitives and the parameters needed for data transfer.

13.1.2.1 *TS User-Data*

The TS User-Data parameter is a TSDU. A TSDU consists of an integral number of octets greater than zero.

TABLE 6/X.214

Data Transfer Primitives and Parameters

Parameter \ Primitive	T-DATA request	T-DATA indication
TS user-data	X	X(=)

X: mandatory parameter.
(=): the value of that parameter is identical to the value of the corresponding parameter of the preceding TS primitive.

13.1.3 *Sequence of TS primitives*

The operation of the TS provider in transferring TS user-data can be modelled as a queue of unknown size within the TS provider (see § 9). The ability of a TS user to issue a T-DATA request depends on the state of the queue. The ability of the TS provider to issue a T-DATA indication depends on the receiving TS user.

The sequence of TS primitives in a successful data transfer is defined in the following time sequence diagram (Figure 7/X.214).

13.2 *Expedited data transfer service*

13.2.1 *Function*

The expedited data transfer service provides a further means of information exchange on a TC in both directions simultaneously. The transfer of expedited TSDUs is subject to different QOS and separate flow control from that applying to TS user-data of the data transfer service.

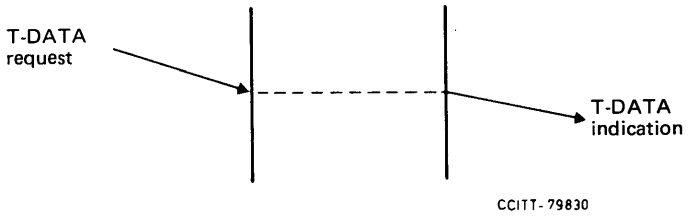


FIGURE 7/X.214

Sequence of Primitives in Data Transfer

The TS provider guarantees that an expedited TSDU will not be delivered after any subsequently submitted normal TSDU or expedited TSDU on that TC.

The relationship between normal and expedited data flow is modelled by the operation of reordering within queues as described in § 9. In particular expedited data will be delivered when the receiving TS user is not accepting normal data. However, the amount of normal data bypassed by such reordering cannot be predicted.

13.2.2 *Types of TS primitives and parameters*

Table 7/X.214 indicates the types of TS primitives and the parameters needed for expedited data transfer.

TABLE 7/X.214

Expedited TS Primitives and Parameters

Parameter \ Primitive	T-EXPEDITED DATA request	T-EXPEDITED DATA indication
TS user-data	X	X(=)

X: mandatory parameter.
(=): the value of the parameter is identical to the value of the corresponding parameter of the preceding TS primitive.

13.2.2.1 *TS User-Data*

The TS User-Data parameter is an expedited TSDU. An expedited TSDU consists of an integral number of octets between 1 and 16 inclusive.

13.2.3 *Sequence of TS primitives*

The sequence of TS primitives in a successful expedited data transfer is defined in the following time sequence diagram (Figure 8/X.214).

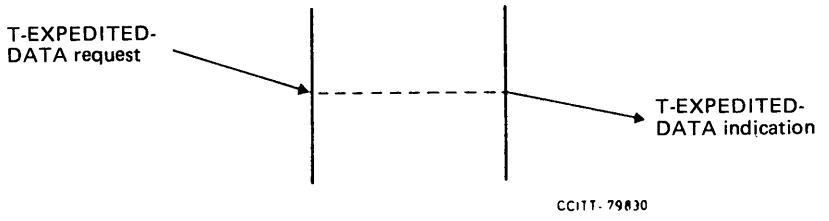


FIGURE 8/X.214

Sequence of Primitives in Expedited Data Transfer

Note – Use of the expedited data transfer service must be requested by the calling TS user and agreed to by the called TS user when the TC is established (see § 12.2.5).

14 Transport Connection release phase

14.1 *Function*

The TC release TS primitives are used to release a TC. The release may be performed:

a) by either or both of the TS users to release an established TC;

b) by the TS provider to release an established TC; all failures to maintain a TC are indicated in this way;

c) by either or both the TS users to abandon TC establishment;

d) by the TS provider, to indicate its inability to establish a requested TC.

TC release is permitted at any time regardless of the current TC phase. A request for release cannot be rejected. The Transport Service does not guarantee delivery of any TS user-data once the release phase in entered.

14.2 *Types of TS primitives and parameters*

Table 8/X.214 indicates the types of TS primitives and the parameters needed for TC release.

14.2.1 *Reason*

The Reason parameter gives information indicating the cause of the TC release. The reason is one of the following:

a) remote TS user invoked;

Note – Additional information may be given in the TS user-data parameter.

b) TS provider invoked. This reason may be of transient or permanent nature.

Note – The following examples are given:

a) lack of local or remote resources of the TS provider,

b) QOS below minimum level,

c) misbehaviour of TS provider,

d) called TS user unknown,

e) called TS user unavailable,

f) unknown reason.

TABLE 8/X.214

TC Release Primitives and Parameters

Parameter \ Primitive	T-DISCONNECT request	T-DISCONNECT indication
Reason		X
TS user-data	X(U)	X(=)

X: mandatory parameter.
(=): the value of that parameter is identical to the value of the corresponding parameter of the preceding TS primitive.
(U): use of this parameter is a TS user option.

14.2.2 *TS User-Data*

The TS user-data parameter allows the transfer of TS user-data between TS users, without modification by the TS provider. The TS user-data may be lost in particular if the TS provider initiates TC release before the T-DISCONNECT indication is delivered, or if both TS users initiate a T-DISCONNECT simultaneously. Therefore, the parameter is only present if the TC release was originated by a TS user. The TS user-data parameter, if present, shall be an integral number of octets in length between 1 and 64 inclusive.

Note 1 – The TS provider may provide additional information (e.g. accounting) for management purposes.

Note 2 – The QOS associated with the TS user-data on the T-DISCONNECT primitives may be lower than the QOS for TS user-data transferred by the T-DATA primitive. TS user-data may be lost without any notice to the TS user receiving the T-DISCONNECT indication, even when initiated by the remote TS user.

14.3 *Sequence of TS primitives when releasing an established transport connection*

The sequence of TS primitives depends on the origin or origins of the TC release action. The sequence may be:

a) invoked by one TS user, with a T-DISCONNECT request from that TS user leading to a T-DISCONNECT indication to the other;

b) invoked by both TS users, with a T-DISCONNECT request from each of the TS users;

c) invoked by the TS provider, with a T-DISCONNECT indication to each of the TS users;

d) invoked independently by one TS user and the TS provider, with a T-DISCONNECT request from the initiating TS user and a T-DISCONNECT indication to the other.

The sequence of TS primitives in these four cases are expressed in the following time sequence diagrams (Figures 9-12/X.214).

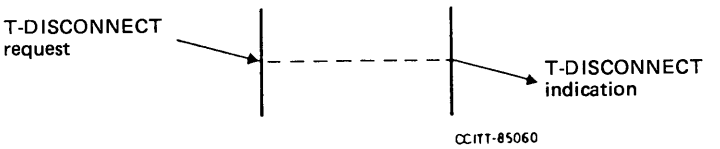


FIGURE 9/X.214

Sequence of Primitives in TS User Invoked Release

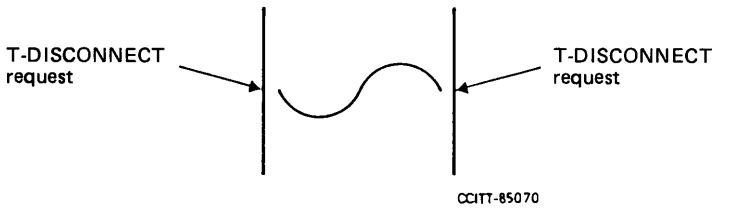


FIGURE 10/X.214

Sequence of Primitives in Simultaneous TS User Invoked Release

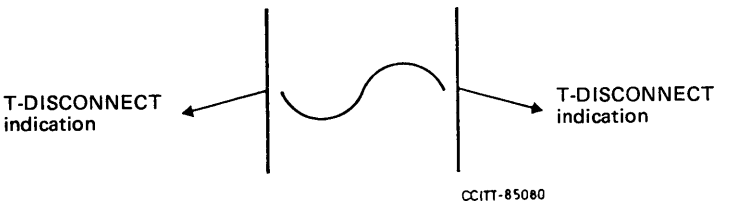


FIGURE 11/X.214

Sequence of Primitives in a TS Provider Invoked Release

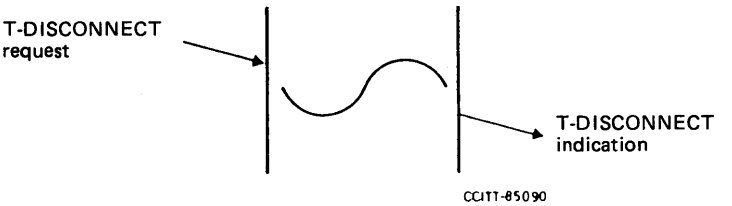


FIGURE 12/X.214

Sequence of Primitives in a Simultaneous TS User and TS Provider Invoked Release

14.4 *Sequence of TS primitives in TS user rejection of a TC establishment*

A TS user may reject a TC establishment attempt by a T-DISCONNECT request. In the T-DISCONNECT indication the reason parameter will indicate that the called TS user initiated the disconnection. The sequence of events is defined in the following time sequence diagram (see Figure 13/X.214).

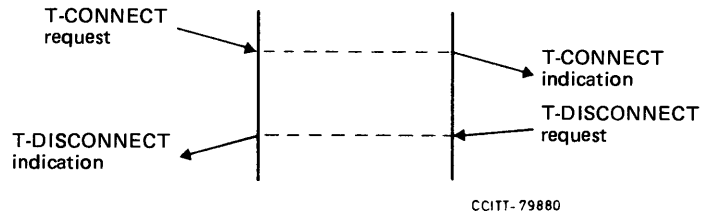


FIGURE 13/X.214

Sequence of Primitives in a TS User rejection of a TC Establishment Attempt

14.5 *Sequence of TS primitives in a TS provider rejection of a TC Establishment attempt*

If the TS provider is unable to establish a TC, it indicates this to the calling TS user by a T-DISCONNECT indication. The reason parameter indicates that the TS provider is the source of the T-DISCONNECT indication. The sequence of events is defined in the following time sequence diagram (Figure 14/X.214).

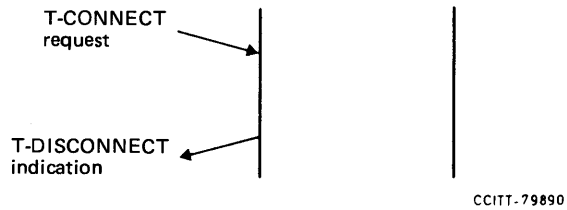


FIGURE 14/X.214

Sequence of Primitives in a TS Provider rejection of a TC Establishment Attempt

15 Formal specification of the Transport Service

(For further study.)

SESSION SERVICE DEFINITION 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.225 specifies the Session Protocol Specification 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 Service of Open Systems Interconnection for CCITT Applications as given in the Scope and Field of Application.

CONTENTS

[1] Recommendation X.215 is technically aligned with ISO 8326 [Information Processing Systems – Open Systems Interconnection – Basic Connection oriented session service definition] which includes corrections resulting from ISO defect reports numbered 8326/4, 8326/6, 8326/9, 8326/11 through 8326/17, 8326/20, 8327/5 and 8327/35 and the addendum 2 to incorporate unlimited user data.

0 Introduction

This Recommendation is one of a set of Recommendations produced to facilitate the interconnection of computer systems. It is related to other Recommendations in the set as defined by the Reference Model for Open Systems Interconnection. The Reference Model subdivides the area of standardization for interconnection into a series of layers of specification, each of manageable size.

The purpose of this Recommendation is to define the service provided to the Presentation Layer at the boundary between the Session and Presentation Layers of the Reference Model. The session service is provided by the session protocol making use of the services available from the Transport Layer. This Recommendation also defines the session service characteristics which the presentation protocol may exploit. The relationship between the Recommendations for the session service, session protocol, transport service, and the presentation protocol is illustrated in Figure 1/X.215.

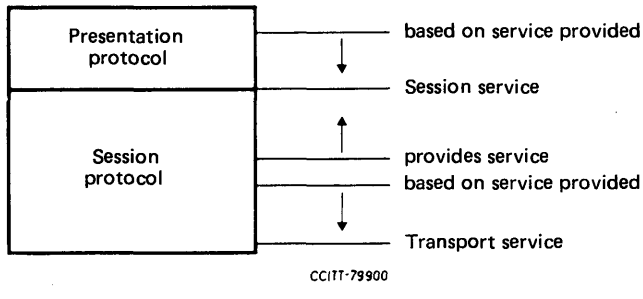


FIGURE 1/X.215

Relationship of this Recommendation to other OSI Recommendations

It is recognized that, with respect to session Quality of Service, (described in § 10), 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, an addendum may be added to this Recommendation at a later time which reflects further QOS developments and integration.

1 Scope and field of application

This Recommendation defines in an abstract way the externally visible service provided by the OSI Session Layer in terms of:

a) the primitive actions and events of the service;

b) the parameter data associated with each primitive action and event;

c) the relationship between, and the valid sequence of these actions and events.

The service defined in this Recommendation is that which is provided by the OSI session protocol (in conjunction with the transport service) and which may be used by any OSI presentation protocol.

This Recommendation does not specify individual implementations or products, nor does it constrain the implementation of entities and interfaces within a computer system. There is, therefore, no conformance to this Recommendation.

2 References

Recommendation X.200	Reference Model of Open Systems Interconnection for CCITT Applications. (See also ISO 1498-1).
Recommendation X.210	OSI Layer Service Definition Conventions. (See also ISO TR8509).
Recommendation X.214	Transport Service Definition for Open Systems Interconnection for CCITT Applications. (See also ISO 8072).
Recommendation X.225	Basic Connection Oriented Session Protocol Specification for Open Systems Interconnection for CCITT Applications. (See also ISO 8327 and ISO 8327 Addendum 2).
ISO 7498-3	Information processing systems – Open Systems Interconnection – Basic Reference Model – Part 3: Naming and Addressing[2)]

3 Definitions

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

3.1 *Reference Model definitions*

This Recommendation is based on the concepts developed in the Basic Reference Model for Open Systems Interconnection (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 service;

e) session-service-access-point;

f) session-service-data-unit;

g) Transport Layer;

h) duplex;

i) half-duplex.

3.2 *Service convention definitions*

This Recommendation also makes use of the following terms defined in the OSI Service Conventions (Recommendation X.210), as they apply to the Session Layer:

a) service-user;

b) service-provider;

c) primitive;

[2)] At present at the stage of draft; publication anticipated in due crouse.

d) request;

e) indication;

f) response;

g) confirm.

3.3 *Session service definitions*

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

3.3.1 **calling SS-user**

An SS-user that initiates a session connection establishment request.

3.3.2 **called SS-user**

An SS-user with whom a calling SS-user wishes to establish a session connection.

Note – Calling SS-users and called SS-users are defined with respect to a single connection. An SS-user can be both a calling and called SS-user simultaneously.

3.3.3 **sending SS-user**

An SS-user that acts as a source of data during the data transfer phase of a session connection.

3.3.4 **receiving SS-user**

An SS-user that acts as a sink of data during the data phase of a session connection.

Note – An SS-user can be both a sending and a receiving SS-user simultaneously.

3.3.5 **requestor; requesting SS-user**

An SS-user that initiates a particular action.

3.3.6 **acceptor; accepting SS-user**

An SS-user that accepts a particular action.

3.3.7 **token**

An attribute of a session connection which is dynamically assigned to one SS-user at a time to permit certain services to be invoked.

3.3.8 **conditional (parameter)**

A parameter whose presence in a request or response depends on conditions defined in the text of this Recommendation; and whose presence in an indication or confirm is mandatory if that parameter was present in the preceding session service primitive, or absent if that parameter was absent in the preceding session service primitive.

3.3.9 **proposed parameter**

The value for a parameter proposed by an SS-user in an S-CONNECT request or an S-CONNECT response that it wishes to use on the session connection.

3.3.10 **selected parameter**

The value for a parameter that has been chosen for use on the session connection.

4 Symbols and abbreviations

4.1. *Abbreviations*

SS: session service

SSAP: session service access point

SSDU: session service data unit

NSSDU: normal data session service data unit

TSSDU: typed data session service data unit

XSSDU: expedited session service data unit

QOS: quality of service

4.2 *Service variables*

V(A): See § 11.4.1.1.

V(M): See § 11.4.1.2.

V(R): See § 11.4.1.3.

Vsc: See § 11.4.1.4.

5 Conventions

This Recommendation uses the descriptive conventions contained in the OSI Service Conventions (Recommendation X.210) except that, where indicated in this Recommendation, parameter values associated with a service primitive may be passed in a direction opposite to the direction of the service primitive.

6 Model of the session service

This Recommendation uses the abstract model for a layer service defined in the OSI Service Conventions (Recommendation X.210). The model defines the interactions between the SS-user and the SS-provider which take place at the two SSAPs. Information is passed between an SS-user and the SS-provider by service primitives, which may convey parameters.

7 Overview of the session service

7.1 *General overview*

The session service provides the means for organized and synchronized exchange of data between cooperating SS-users. It provides its users with means to:

a) establish a connection with another SS-user, exchange data with that user in a synchronized manner, and release the connection in an orderly manner;

b) negotiate for the use of tokens to exchange data, synchronize and release the connection, and to arrange for data exchange to be half-duplex or duplex;

c) establish synchronization points within the dialogue and, in the event of errors, resume the dialogue from an agreed synchronization point;

d) interrupt a dialogue and resume it later at a prearranged point.

7.2 *Token concept*

A token is an attribute of a session connection which is dynamically assigned to one SS-user at a time to permit certain services to be invoked. It is the right to exclusive use of the service.

Four tokens are defined:

a) the data token;

b) the release token;

c) the synchronize-minor token;

d) the major/activity token.

A token is always in one of the following states:

e) *available*, in which case it is always:

1) *assigned* to one SS-user, who then has the exclusive right to use the associated service (provided that no other restrictions apply); and

2) *not assigned* to the other SS-user, who does not have the right to use the service but may acquire it later; or

f) *not available* to either SS-user, in which case neither SS-user has the exclusive use of the associated service. The service then becomes inherently available to both SS-users (data transfer and release), or otherwise unavailable to both SS-users (synchronization and activities).

Restrictions related to the availability and assignment of tokens are defined in § 11.2.

7.3 *Synchronization and dialogue unit concepts*

SS-users may insert synchronization points into the data they are transmitting. Each synchronization point is identified by a serial number maintained by the SS-provider (see § 11.4).

Any semantics which SS-users may give to their synchronization points are transparent to the SS-provider.

There are two types of synchronization points:

a) minor synchronization points;

b) major synchronization points.

Major synchronization points are used to structure the exchange of data into a series of dialogue units. The characteristic of a dialogue unit is that all communication within it is completely separated from all communication before and after it. A major synchronization point indicates the end of one dialogue unit and the beginning of the next. Each major synchronization point is confirmed explicitly.

Minor synchronization points are used to structure the exchange of data within a dialogue unit. Figure 2/X.215 illustrates how a dialogue unit is structured through the use of minor synchronization points. Each minor synchronization point may or may not be confirmed explicitly.

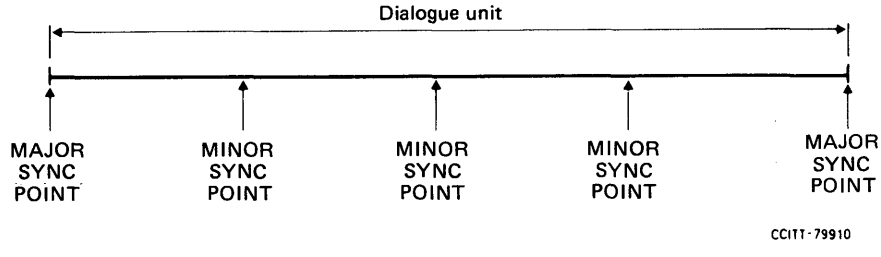


FIGURE 2/X.215

Example of a structured dialogue unit

7.4 *Activity concept*

The activity concept allows SS-users to distinguish between different logical pieces of work called activities. Each activity consists of one or more dialogue units. Only one activity is allowed on a session connection at a time, but there may be several consecutive activities during a session connection. An activity may also span more than one session connection. An activity can be interrupted and then resumed on the same or on a subsequent session connection. This can be considered as a form of resynchronization.

Figure 3/X.215 shows how an activity may be structured into dialogue units through the use of major synchronization points. In addition, the SS-users may transfer data outside an activity.

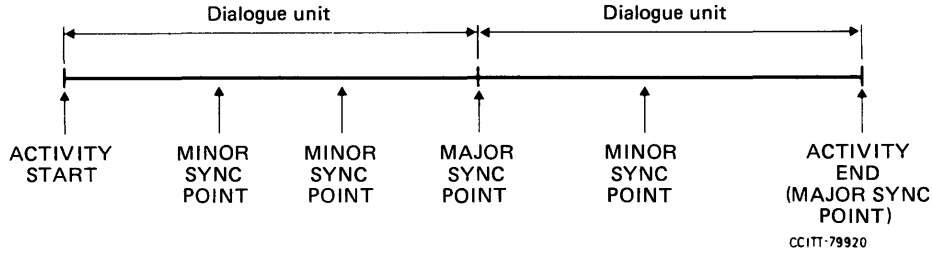


FIGURE 3/X.215

Example of a structured activity

7.5 *Resynchronization*

Resynchronization may be initiated by either SS-user. It sets the session connection to a defined state, and therefore includes reassignment of tokens and setting the synchronization point serial number to a new value. Resynchronization purges all undelivered data.

Three options are provided:

a) *abandon option* which is used to set the synchronization point serial number to an unused value;

b) *restart option* which is used to set the synchronization point serial number to any used value which is greater than the synchronization point serial number which identifies the last acknowledged major synchronization point;

c) *set option* which is used to set the synchronization point serial number to any value chosen by SS-user.

7.6 *Negotiation*

Negotiation takes place between both SS-users during the session connection establishment phase according to the following rules.

7.6.1 *Negotiation of functional units*

The kernel functional unit (see § 9) is always used. Each SS-user proposes the use or non-use of each of the other functional units. A functional unit is selected only if both SS-users propose use of the functional unit and it is supported by the SS-provider. Specific negotiation rules are given in § 12.1.2.

7.6.2 *Negotiation of initial token settings*

When the calling SS-user proposes use of a functional unit that requires a token, it also proposes the initial token settings:

a) calling SS-user side;

b) called SS-user side;

c) called SS-user choice.

If the use of the functional unit is selected, the token is set to:

d) the side proposed by the called SS-user, if "called SS-user choice" is proposed by the calling SS-user; or

e) in all other cases, the side proposed by the calling SS-user.

7.6.3 *Negotiation of initial synchronization point serial number*

When a calling SS-user proposes any of the major synchronize, minor synchronize or resynchronize functional units, but does not propose the activity management functional unit, it also proposes a value for the initial synchronization point serial number. The calling SS-user may also propose a value for the initial synchronization point serial number 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. If the called SS-user selects use of any of the minor synchronize, major synchronize or resynchronize functional units, but does not select use of the activity management functional unit, it returns a value for the initial synchronization point serial number which may or may not be the same as the value proposed by the calling SS-user. The value returned by the called SS-user is used as the initial synchronization point serial number for the session connection.

In all other combinations of functional units, no initial synchronization point serial number is proposed.

8 Phases and services of the session service

The session service comprises three phases. The purpose of each phase, and a short description of the associated services is given in this section. The services and the primitives by which they are invoked are defined in §§ 12, 13 and 14.

Note – The amount of SS-user data which can be transferred in certain primitives may be limited due to protocol restrictions imposed by the SS-provider.

8.1 *Session connection establishment phase*

The session connection establishment phase is concerned with establishing a connection between two SS-users. It has one service associated with it:

The *Session Connection* service (see § 12.1) is used to set up a session connection and to negotiate tokens and parameters to be used for the connection.

8.2 *Data transfer phase*

The data transfer phase is concerned with the exchange of data between the two SS-users connected in the session connection establishment phase.

There are four services associated with data transfer:

a) the *Normal Data Transfer* service (see § 13.1) allows the transfer of normal data SSDUs (NSSDUs) over a session connection. Its use is controlled by the data token if the half-duplex functional unit has been selected;

b) the *Expedited Data Transfer* service (see § 13.2) allows the transfer of expedited SSDUs (XSSDUs) over a session connection free from the token and flow control constraints of the Normal Data Transfer service, Typed Data Transfer service and Capability Data Exchange service;

c) the *Typed Data Transfer* service (see § 13.3) is used to transfer typed data SSDUs (TSSDUs) independent of the availability and assignment of the data token;

d) the *Capability Data Exchange* service (see § 13.4) is used to exchange confirmed SS-user data while not within an activity.

There are three services concerned with token management:

e) the *Give Tokens* service (see § 13.5) allows an SS-user to surrender one or more specific tokens to the other SS-user;

f) the *Please Tokens* service (see § 13.6) allows an SS-user to request the other SS-user to transfer one or more specific tokens to it;

g) the *Give Control* service (see § 13.7) allows an SS-user to surrender all available tokens to the other SS-user.

There are three services associated with synchronization and resynchronization:

h) the *Minor Synchronization Point* service (see § 13.8) allows the SS-user to separate the flow of NSSDUs and TSSDUs transmitted before the service was invoked from the subsequent flow of NSSDUs and TSSDUs. Its use is controlled by the synchronize-minor token;

i) the *Major Synchronization Point* service (see § 13.9) allows the SS-user to confine the flow of sequentially transmitted NSSDUs, TSSDUs and XSSDUs in each direction within a dialogue unit. Its use is controlled by the major/activity token;

j) the *Resynchronize* service (see § 13.10) is used to set the session connection to a previous or to a new synchronization point and to reassign the available tokens. This service may cause loss of NSSDUs, TSSDUs and XSSDUs.

There are two services for reporting errors or unanticipated situations:

k) the *Provider-Initiated Exception Reporting* service (see § 13.11) (P-Exception Reporting service) permits SS-users to be notified of exception conditions or SS-provider protocol errors. This service may cause loss of NSSDUs, TSSDUs and XSSDUs;

l) the *User-Initiated Exception Reporting* service (see § 13.12) (U-Exception Reporting service) is used by the SS-user to report an exception condition when the data token is available but not assigned to the SS-user. This service may cause loss of NSSDUs, TSSDUs and XSSDUs.

There are five services associated with activities:

m) the *Activity Start* service (see § 13.13) is used to indicate that a new activity is entered. Its use is controlled by the major/activity token;

n) the *Activity Resume* service (see § 13.14) is used to indicate that a previously interrupted activity is re-entered. Its use is controlled by the major/activity token;

o) the *Activity Interrupt* service (see § 13.15) allows an activity to be abnormally terminated with the implication that the work so far achieved is not to be discarded and may be resumed later. Its use is controlled by the major/activity token. This service may cause loss of NSSDUs, TSSDUs and XSSDUs;

p) the *Activity Discard* service (see § 13.16) allows an activity to be abnormally terminated with the implication that the work so far achieved is to be discarded, and not resumed. Its use is controlled by the major/activity token. This service may cause loss of NSSDUs, TSSDUs and XSSDUs;

q) the *Activity End* service (see § 13.17) is used to end an activity (and set a major synchronization point). Its use is controlled by the major/activity token.

Using the activity services may lead to a state where no activity is in progress on the session connection. When activity services are employed, but no activity is in progress, only the activity start, activity resume, token management, capability data, typed data, normal data, expedited data, abort and release services may be invoked by the SS-users.

8.3 *Session connection release phase*

The session connection release phase is concerned with releasing a previously established session connection. It has three services associated with it:

a) the *Orderly Release* service (see § 14.1) provides a means of achieving the orderly release of a session connection;

b) the *User-Initiated Abort* service (see § 14.2) (U-Abort service) is used to initiate the release of a session connection in a way that will terminate any outstanding service request. This service may cause loss of NSSDUs, TSSDUs and XSSDUs;

c) the *Provider-Initiated Abort* service (see § 14.3) (P-Abort service) is used by the SS-provider to indicate the release of the session connection for internal reasons. This service may cause loss of NSSDUs, TSSDUs and XSSDUs. Any outstanding service request is terminated.

9 Functional units and subsets

9.1 *Functional units*

Functional units are logical groupings of related services defined by this Recommendation for the purpose of:

a) negotiation of SS-user requirements during the session connection establishment phase;

b) reference by other Recommendations.

Table 1/X.215 specifies the association of tokens and functional units. When a functional unit implies the availability of a token, services concerned with the management of that token are provided in order to be able to request and transfer the available tokens.

The services associated with each functional unit are specified in Table 2/X.215.

9.1.1 *Kernel functional unit*

The kernel functional unit supports the basic session services required to establish a session connection, transfer normal data and release the session connection.

9.1.2 *Negotiated release functional unit*

The negotiated release functional unit supports the negotiated orderly release service. The release token is available when this functional unit has been selected.

9.1.3 *Half-duplex functional unit*

The half-duplex functional unit supports the half-duplex service. The data token is available when this functional unit has been selected. It is not possible to select both this functional unit and the duplex functional unit for use on the same session connection.

9.1.4 *Duplex functional unit*

The duplex functional unit supports the duplex service. It is not possible to select both this functional unit and the half-duplex functional unit for use on the same session connection.

9.1.5 *Expedited data functional unit*

The expedited data functional unit supports the session expedited data transfer service.

9.1.6 *Typed data functional unit*

The typed data functional unit supports the typed data transfer service.

9.1.7 *Capability data exchange functional unit*

The capability data exchange functional unit supports the capability data exchange service. This functional unit can only be selected when the activity management functional unit has been selected.

9.1.8 *Minor synchronize functional unit*

The minor synchronize functional unit supports the minor synchronization point service. The synchronize-minor token is available when this functional unit has been selected.

9.1.9 *Major synchronize functional unit*

The major synchronize functional unit supports the major synchronization point service. The major/activity token is available when this functional unit has been selected.

9.1.10 *Resynchronize functional unit*

The resynchronize functional unit supports the resynchronize service.

9.1.11 *Exceptions functional unit*

The exceptions functional unit supports the user exception and provider exception reporting services.

This functional unit can only be selected when the half-duplex functional unit has been selected.

9.1.12 *Activity management functional unit*

The activity management functional unit supports the activity management services and the give control service. The major/activity token is available when this functional unit has been selected.

9.2 *Subsets*

A subset is defined as a combination of the kernel functional unit together with any other set of functional units provided that:

a) if the capability data functional unit is included in the subset, then the activity management functional unit is also included in the subset; and

b) if the exceptions functional unit is included in the subset, then the half duplex functional unit is also included in the subset.

Note – This Recommendation contains no requirements for the registration of subsets. Users of this Recommendation may define subsets to meet their session service needs. Other Recommendations may identify subsets that comply with the above definition.

TABLE 1/X.215

Functional Units Using Tokens

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 2/X.215

Services Associated With Each Functional Unit

Functional unit	Service(s)	Reference
Kernel (non-negotiable)	Session connection Normal data transfer Orderly release U-Abort P-Abort	12.1 13.1 14.1 14.2 14.3
Negotiated release	Orderly release Give tokens Please tokens	14.1 13.5 13.6
Half-duplex	Give tokens Please tokens	13.5 13.6
Duplex	No additional service	
Expedited data	Expedited data transfer	13.2
Typed data	Typed data transfer	13.3
Capability data exchange	Capability data exchange	13.4
Minor synchronize	Minor synchronization point Give tokens Please tokens	13.8 13.5 13.6
Major synchronize	Major synchronization point Give tokens Please tokens	13.9 13.5 13.6
Resynchronize	Resynchronize	13.10
Exceptions	Provider exception reporting User exception reporting	13.11 13.12
Activity management	Activity start Activity resume Activity interrupt Activity discard Activity end Give tokens Please tokens Give control	13.13 13.14 13.15 13.16 13.17 13.5 13.6 13.7

10 Quality of session service

The term "quality of service" (QOS) refers to certain characteristics of a session connection as observed between the session connection endpoints. QOS describes aspects of a session connection which are attributable solely to the SS-provider; such aspects are independent of SS-user behaviour (which is beyond the control of the SS-provider). SS-user behaviour does not impact the QOS provided.

Once a session connection is established, the SS-users at the two ends have the same knowledge and understanding of what the QOS over the session connection is.

10.1 *Determination of QOS*

QOS is described in terms of QOS parameters.

The definition of the QOS parameters associated with the session service is given in § 10.3. These definitions provide both SS-users and the SS-provider with a common understanding of the QOS characteristics.

Two types of session service QOS parameters are identified:

a) those which are negotiated during the session connection establishment phase:
   1) session connection protection (see § 10.3.9);
   2) session connection priority (see § 10.3.10);
   3) residual error rate (see § 10.3.5);
   4) throughput, for each direction of transfer (see § 10.3.3);
   5) transit delay, for each direction of transfer (see § 10.3.4);
   6) optimized dialogue transfer (see § 10.3.13); and
   7) extended control (see § 10.3.12);

b) those which are not negotiated during the session connection establishment phase but whose values are selected and/or known by other methods (for example, *a priori* knowledge and agreement, or by means of management functions) not defined in this Recommendation:
   1) session connection establishment delay (see § 10.3.1);
   2) session connection establishment failure probability (see § 10.3.2);
   3) transfer failure probability (see § 10.3.6);
   4) session connection release delay (see § 10.3.7);
   5) session connection release failure probability (see § 10.3.8);
   6) session connection resilience (see § 10.3.11).

The negotiation procedures for parameters listed in § 10.1 a) are defined in § 10.2. Once the session connection is established, the selected QOS parameters are not renegotiated during the lifetime of the session connection. The SS-user should be aware that changes in QOS during a session connection are not signalled in the session service.

10.2 *Session connection QOS negotiation procedures*

QOS negotiation is described in terms of parameters which can be conveyed by the S-CONNECT primitives during the session connection establishment phase (see § 12). For the parameters which are negotiated during the session connection establishment phase [see § 10.1 a)], the parameter values and negotiation rules are defined as follows:

a) In the S-CONNECT request primitive, the calling SS-user can specify:
   1) for session connection protection, session connection priority, extended control, and optimized dialogue transfer, a single parameter value which is the "desired" QOS; for extended control and optimized dialogue transfer, one of the two values "feature desired" or "feature not desired" is conveyed;

      Note – If the calling SS-user proposes use of the expedited data functional unit, the extended control parameter has the value "feature desired".

   2) for residual error rate, and for each direction of throughput and transit delay, two parameter values which are the "desired" QOS and the "lowest acceptable" QOS to which the calling SS-user will agree.

b) In the S-CONNECT indication primitive, for each of the negotiated parameters, an "available" value is conveyed which is specified as follows:

1) for session connection protection, if the SS-provider agrees to provide a QOS value equivalent to the "desired" value specified in the S-CONNECT request, then the SS-provider specifies that value as "available"; if the SS-provider does not agree to provide the "desired" QOS requested, the SS-provider refuses to establish the session connection by issuing the S-CONNECT (reject) confirm primitive to the calling SS-user;

2) for session connection priority, the SS-provider specifies the QOS value it is willing to provide (a value which is equal to or better than the "desired" value specified in the S-CONNECT request) as "available";

3) for the residual error rate and each direction of throughput and transit delay, if the SS-provider agrees to provide a value of QOS which is equal to or better than the "lowest acceptable" QOS value specified in the S-CONNECT request, then the SS-provider specifies the value as "available"; if the SS-provider does not agree to provide this QOS, then the SS-provider refuses to establish the session connection by issuing the S-CONNECT (reject) confirm primitive to the calling SS-user;

4) for extended control and optimized dialogue transfer, if the "desired" value in the S-CONNECT request primitive is "feature not desired" then "feature not desired" is specified as "available"; if the "desired" value is "feature desired" and the SS-provider agrees to provide the feature on the session connection, then "feature desired" is specified as "available"; otherwise if the SS-provider does not agree to provide the feature, "feature not desired" is specified as "available".

c) In the S-CONNECT response primitive, for each of the negotiated parameters, an "agreed" value is conveyed which is specified as follows:

1) for optimized dialogue transfer, if the "available" value in the S-CONNECT indication primitive is "feature not desired" and the called SS-user agrees not to have the feature provided on the session connection, then "feature not desired" is specified as "agreed"; otherwise the SS-user may reject establishment of the session connection; if the "available" value in the indication primitive is "feature desired" and the SS-user agrees to have the feature provided, then "feature desired" is specified as "agreed"; otherwise, if the SS-user does not agree to provision of the feature, the value "feature not desired" is specified as "agreed";

2) for each of the other parameters, if the called SS-user agrees to the QOS value specified as "available" in the S-CONNECT indication primitive, then the identical value is specified as "agreed"; if the SS-user does not agree to the "available" value, the SS-user may reject establishment of the session connection.

d) In the S-CONNECT confirm primitive, for each of the negotiated parameters, an "agreed" value is conveyed which is identical to the "agreed" value conveyed in the S-CONNECT response.

10.3 *Definition of QOS parameters*

QOS parameters can be classified as:

a) parameters which express session service performance parameters, as shown in Table 3/X.215;

b) parameters which express other session service characteristics, as shown in Table 4/X.215.

These session service QOS parameters are defined in this subsection.

10.3.1 *Session connection establishment delay*

Session connection establishment delay is the maximum acceptable delay between an S-CONNECT request and the corresponding S-CONNECT confirm primitive.

Note – This delay includes SS-user dependent components.

10.3.2 *Session connection establishment failure probability*

Session connection establishment failure probability is the ratio of total session connection establishment failures to total session connection establishment attempts in a measurement sample.

TABLE 3/X.215

Classification of Performance QOS Parameters

Phase	Performance criterion	
	Speed	Accuracy/Reliability
Session connection establishment	Session connection establishment delay	Session connection establishment failure probability (misconnection/session connection refusal)
Data transfer	Throughput transit delay	Residual error rate (corruption, duplication/loss) Session connection resilience Transfer failure probability
Session connection release	Session connection release delay	Session connection release failure probability

TABLE 4/X.215

Parameters Specifying Other Session Service Features

Extended control
Session connection protection
Session connection priority
Optimized dialogue transfer

Session connection establishment failure is defined to occur when a requested session connection is not established within the specified maximum acceptable session connection establishment delay as a result of misconnection, session connection refusal, or excessive delay on the part of the SS-provider. Session connection establishment attempts which fail as a result of error, session connection refusal, or excessive delay on the part of an SS-user are excluded in calculating session connection establishment failure probability.

10.3.3 *Throughput*

Throughput is defined for each direction of transfer, in terms of a sequence of at least two SSDUs successfully transferred by an S-DATA request/S-DATA indication or S-TYPED-DATA request/S-TYPED-DATA indication sequence of primitives. Given such a sequence of n SSDUs, where n is greater than or equal to two, the throughput is defined to be the smaller of:

a) the number of SS-user data octets contained in the last $n - 1$ SSDUs divided by the time between the first and last S-DATA or S-TYPED-DATA request in the sequence; and

b) the number of SS-user data octets contained in the last $n - 1$ SSDUs divided by the time between the first and the last S-DATA or S-TYPED-DATA indications in the sequence.

Successful transfer of the octets in a transmitted SSDU is defined to occur when the bits are delivered to the intended receiving SS-user without error, in the proper sequence, prior to release of the session connection by the receiving SS-user.

Throughput is only meaningful for a sequence of complete SSDUs and each specification is based on a previously stated average SSDU size.

Throughput is specified separately for each direction of transfer on a session connection. In each direction, a specification of throughput will consist of a "maximum throughput" value and an "average throughput" value. The "maximum throughput" value represents the maximum rate at which the SS-provider can continuously accept and deliver SSDUs, in the absence of sending SS-user input delays or flow control applied by the receiving SS-user. Thus, the sequence of SSDUs in the calculation above are defined to be presented continuously at the maximum rate. The "average throughput" value represents the expected transfer rate on a session connection including the effects of expected user-attributable delays (e.g. non-continuous SSDU input, receiving SS-user flow control). Thus, the sequence of SSDUs in the calculation above are defined to be presented at a rate which includes components representing "average" user delays.

It is possible for either the input or the output of a sequence of SSDUs to be excessively delayed by the SS-users. Such occurrences are excluded in calculating "average throughput" values.

For each direction of transfer, and for each of the "maximum throughput" and "average throughput" specifications, the throughput QOS for a particular session connection is negotiated between the SS-users and the SS-provider (see § 10.2).

Throughput on a session connection relates only to the transfer of normal data and typed data over the session connection. There is no specification of the throughput for data which is transferred in association with the issue of any other session service primitives (e.g. S-CONNECT, S-CAPABILITY-DATA, etc.).

10.3.4 *Transit delay*

Transit delay is the elapsed time between the completion of any session service request primitive and the corresponding session service indication primitive occurring during the data transfer phase of a session connection. Elapsed time values are calculated only on service primitive pairs which are successfully completed.

Successful completion of a service primitive pair is defined to occur when the issue of the request primitive by one SS-user results in the issue of the corresponding indication primitive to the peer user (including any SS-user data associated with the primitive) which is without error, and in a proper sequence with respect to other primitives, prior to release of the session connection by the receiving SS-user.

In duplex and half-duplex session connections, transit delay is specified independently for each direction of transfer. In general, each transit delay specification defines both the average value and the maximum value expected for a session connection. Each specification of transit delay assumes a previously stated average size for SS-user data included in the service primitive pair.

An attempt to measure the transit delay for an individual service primitive pair may be greatly influenced if the receiving SS-user exercises flow control. Such occurrences are excluded in calculating both average and maximum transit delay values.

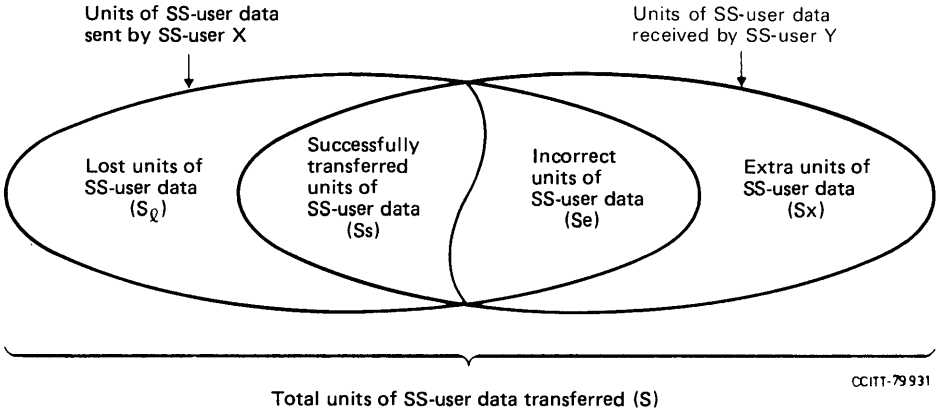
10.3.5 *Residual error rate*

Residual error rate is the ratio of total incorrect, lost, and duplicate units of SS-user data to the total units of SS-user data transferred across the session service boundary in association with any SS-primitive issued in the data transfer phase of a session connection during a measurement period. The relationship between these quantities for a particular SS-user pair is defined in Figure 4/X.215.

10.3.6 *Transfer failure probability*

Transfer failure probability is the ratio of total transfer failures to total transfer samples observed during a performance measurement.

A transfer sample is a discrete observation of SS-provider performance in handling service requests made by the SS-user. A transfer sample begins with the initiation of session service requests during the data transfer phase and continues until the outcome of a given number of service requests have been determined. These service requests may include the transfer of SS-user data or other service requests (such as S-ACTIVITY-START request, S-TOKEN-PLEASE request, etc.) made by the SS-user. A transfer sample will normally correspond to the duration of an individual session connection.



$$RER = \frac{Se + S_Q + Sx}{S}$$

FIGURE 4/X.215

Components of residual error rate

A transfer failure is a transfer sample in which the observed performance is worse than a specified minimum acceptable level. Transfer failures are identified by comparing the measured values for applicable supported performance parameters with specified transfer failure thresholds. The three supported performance parameters which may apply are throughput, transit delay, and residual error rate.

In systems where session service QOS is reliably monitored by the SS-provider, transfer failure probability can be estimated by the probability of an S-P-ABORT or an S-P-EXCEPTION-REPORT during a transfer sample.

10.3.7 *Session connection release delay*

Session connection release delay is the maximum acceptable delay between an SS-user initiated S-U-ABORT request and the successful release of a particular session connection. Session connection release delay is normally specified independently for each SS-user.

Issue of an S-U-ABORT request by either SS-user marks the beginning of the session connection release delay for both users. Successful release for one SS-user is defined to occur when that SS-user is first able to initiate a new session connection. Successful release is signalled to the SS-user not initiating the S-U-ABORT request by an S-U-ABORT indication. The SS-user initiating the S-U-ABORT request will normally receive a similar signal of local significance.

10.3.8 *Session connection release failure probability*

Session connection release failure probability is the ratio of total SS-user initiated abort requests resulting in session connection release failure to total SS-user initiated abort requests included in a measurement sample. Session connection release failure probability is normally specified independently for each SS-user.

Session connection release failure is defined to occur, for a particular SS-user, if that SS-user is not successfully released (as defined in § 10.3.7) within the specified maximum session connection release delay as a result of error or excessive delay on the part of the SS-provider. Session connection release attempts which fail as a result of error, release refusal, or excessive delay on the part of an SS-user are excluded in calculating session connection release failure probability.

10.3.9 *Session connection protection*

Session connection protection is the extent to which an SS-provider attempts to prevent unauthorized monitoring or manipulation of SS-user originated information. Session connection protection is specified qualitatively by selecting one of the following session connection protection options:

a) no protection features;

b) protection against passive monitoring;

c) protection against modification, replay, addition or deletion;

d) both b) and c).

10.3.10 *Priority*

The specification of priority is concerned with the relationship between session connections. This parameter specifies the relative importance of a session connection with respect to:

a) the order in which session connections are to have their QOS degraded, if necessary; and

b) the order in which session connections are to be broken to recover resources, if necessary.

This parameter only has meaning in the context of some management entity or structure able to judge relative importance. The number of priority levels is limited.

10.3.11 *Session connection resilience*

Session connection resilience parameters specify the probability of:

a) an SS-provider initiated non-orderly release of a session connection (i.e. issue of an S-P-ABORT indication); and

b) an SS-provider exception report (i.e. issue of an S-P-EXCEPTION-REPORT indication) during a specified time interval on an established session connection.

10.3.12 *Extended control parameter*

The extended control parameter allows the SS-users to make use of the resynchronize, abort, activity interrupt and activity discard services when normal flow is congested.

Note – When the expedited data functional unit has been selected, the extended control QOS is always provided to the SS-users.

10.3.13 *Optimized dialogue transfer*

The optimized dialogue transfer QOS parameter permits the concatenated transfer of certain session service requests. How this concatenation of service requests is achieved is a local implementation matter.

Note – This QOS parameter invokes the SS-provider extended concatenation protocol option.

11 Introduction to session service primitives

11.1 *Summary of primitives*

Each of the services constituting the session service is achieved by invoking a sequence of session service primitives. Tables 5/X.215, 6/X.215 and 7/X.215 summarize the primitives and their parameters occurring in each phase of the session service. The parameters are defined in §§ 12, 13 and 14.

TABLE 5/X.215

Session Connection Establishment Phase Primitives

Service	Primitives	Parameters
Session connection	S-CONNECT request	Session connection identifier
	S-CONNECT indication	Calling/Called/Responding session addresses
	S-CONNECT response	Result, QOS, Session requirements
	S-CONNECT confirm	Synchronization point serial number, Initial assignment of tokens, SS-user data.

TABLE 6/X.215

Data Transfer Phase Primitives

Service	Primitives	Parameters
Normal data transfer	S-DATA request S-DATA indication	SS-user data
Expedited data transfer	S-EXPEDITED-DATA request S-EXPEDITED-DATA indication	SS-user data
Typed data transfer	S-TYPED-DATA request S-TYPED-DATA indication	SS-user data
Capability data exchange	S-CAPABILITY-DATA request S-CAPABILITY-DATA indication S-CAPABILITY-DATA response S-CAPABILITY-DATA confirm	SS-user data
Given tokens	S-TOKEN-GIVE request S-TOKEN-GIVE indication	Tokens, SS-user data
Please tokens	S-TOKEN-PLEASE request S-TOKEN-PLEASE indication	Tokens, SS-user data
Give control	S-CONTROL-GIVE request S-CONTROL-GIVE indication	SS-user data
Minor synchronization point	S-SYNC-MINOR request S-SYNC-MINOR indication S-SYNC-MINOR response S-SYNC-MINOR confirm	Type, Synchronization point serial number, SS-user data
Major synchronization point	S-SYNC-MAJOR request S-SYNC-MAJOR indication S-SYNC-MAJOR response S-SYNC-MAJOR confirm	Synchronization point serial number, SS-user data

TABLE 6/X.215 *(cont.)*

Service	Primitives	Parameters
Resynchronize	S-RESYNCHRONIZE request S-RESYNCHRONIZE indication S-RESYNCHRONIZE response S-RESYNCHRONIZE confirm	Resynchronize type, Synchronization point serial number, Assignment of tokens, SS-user data
P-Exception report	S-P-EXCEPTION-REPORT indication	Reason
U-Exception reporting	S-U-EXCEPTION-REPORT request S-U-EXCEPTION-REPORT indication	Reason, SS-user data
Activity start	S-ACTIVITY-START request S-ACTIVITY-START indication	Activity identifier, SS-user data
Activity resume	S-ACTIVITY-RESUME request S-ACTIVITY-RESUME indication	Activity identifier, Old activity identifier, Synchronization point serial number, Old session connection identifier, SS-user data
Activity interrupt	S-ACTIVITY-INTERRUPT request S-ACTIVITY-INTERRUPT indication S-ACTIVITY-INTERRUPT response S-ACTIVITY-INTERRUPT confirm	Reason, SS-user data
Activity discard	S-ACTIVITY-DISCARD request S-ACTIVITY-DISCARD indication S-ACTIVITY-DISCARD response S-ACTIVITY-DISCARD confirm	Reason, SS-user data
Activity end	S-ACTIVITY-END request S-ACTIVITY-END indication S-ACTIVITY-END response S-ACTIVITY-END confirm	Synchronization point serial number, SS-user data

TABLE 7/X.215

Session Connection Release Phase Primitives

Service	Primitives	Parameters
Orderly release	S-RELEASE request S-RELEASE indication S-RELEASE response S-RELEASE confirm	Result, SS-user data
U-Abort	S-U-ABORT request S-U-ABORT indication	SS-user data
P-Abort	S-P-ABORT indication	Reason

11.2 *Token restrictions on sending primitives*

Table 8/X.215 defines the conditions under which those service primitives requiring tokens may be issued.

11.3 *Sequencing of primitives*

All SS-user requests and responses are delivered by the SS-provider in the order in which they are submitted by the SS-user, except for the following:

a) S-EXPEDITED-DATA;

b) S-RESYNCHRONIZE;

c) S-ACTIVITY-INTERRUPT;

d) S-ACTIVITY-DISCARD;

e) S-U-ABORT,

which may be delivered earlier than previously submitted primitives, but not later than subsequently submitted primitives.

11.4 *Synchronization point serial number management*

Certain primitives carry a synchronization point serial number, which is used to identify a synchronization point. Synchronization points are assigned valid synchronization point serial numbers in the range 0 to 999998 by the SS-provider. It is the responsibility of the SS-user to ensure that the number assigned by the SS-provider in a synchronization point request does not exceed 999998.

The synchronization point serial number 999999 is also a valid synchronization point serial number for use by the SS-user, but only in the following services, which require the synchronization point serial number of the next synchronization point:

a) Session Connection Service;

b) Resynchronization Service.

The management of synchronization point serial numbers is defined in this Recommendation in terms of:

a) operations on abstract local variables V(M), V(A), V(R) and Vsc, managed by the SS-provider, and

b) primitives issued by the SS-user in order to invoke these operations.

These operations are summarized in Table A-4/X.215 in Annex A.

TABLE 8/X.215

Token Restrictions on Service Primitives

Service primitives	Data token	Synchronize-minor token	Major/activity token	Release token
S-RELEASE request	2	2	2	2
S-RELEASE response (negative)	nr	nr	nr	0
S-DATA request (half-duplex)	1	nr	nr	nr
S-DATA request (duplex)	3	nr	nr	nr
S-CAPABILITY-DATA request	2	2	1	nr
S-TOKEN-GIVE request (data token)	1	nr	nr	nr
S-TOKEN-GIVE request (synchronize-minor token)	nr	1	nr	nr
S-TOKEN-GIVE request (major/activity token)	nr	nr	1	nr
S-TOKEN-GIVE request (release token)	nr	nr	nr	1
S-TOKEN-PLEASE request (data token)	0	nr	nr	nr
S-TOKEN-PLEASE request (synchronize-minor token)	nr	0	nr	nr
S-TOKEN-PLEASE request (major/activity token)	nr	nr	0	nr
S-TOKEN-PLEASE request (release token)	nr	nr	nr	0
S-CONTROL-GIVE request	2	2	1	2
S-SYNC-MINOR request	2	1	nr	nr
S-SYNC-MAJOR request	2	2	1	nr
S-U-EXCEPTION-REPORT request	0	nr	nr	nr
S-ACTIVITY-START request	2	2	1	nr
S-ACTIVITY-RESUME request	2	2	1	nr
S-ACTIVITY-INTERRUPT request	nr	nr	1	nr
S-ACTIVITY-DISCARD request	nr	nr	1	nr
S-ACTIVITY-END request	2	2	1	nr

0: Token available and not assigned to the SS-user who initiated the service primitive.
1: Token available and assigned to the SS-user who initiated the service primitive.
2: Token not available or token assigned to the SS-user who initiated the service primitive.
3: Token not available.
nr: No restriction.

11.4.1 *Variables*

11.4.1.1 *V(A)*

V(A) is the lowest serial number to which a synchronization point confirmation is expected. No confirmation is expected when V(A) = V(M).

11.4.1.2 *V(M)*

V(M) is the next serial number to be used.

11.4.1.3 *V(R)*

V(R) is the lowest serial number to which resynchronization restart is permitted.

11.4.1.4 *Vsc*

Vsc is used 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.

11.4.2 *Session connection establishment*

When a session connection is established in which at least one of the following functional units has been selected:

a) minor synchronize functional unit; or

b) major synchronize functional unit; or

c) resynchronize functional unit

and the activity management functional unit has not been selected, V(M) and V(A) are set to the initial synchronization point serial number of the response/confirm primitives. V(R) is set to zero. Vsc is set to false.

11.4.3 *Minor synchronization point*

When an S-SYNC-MINOR request is issued, the associated synchronization point serial number, which is indicated to the SS-user, is equal to V(M). V(R) remains unchanged. V(A) is set to V(M) if Vsc is true, otherwise V(A) remains unchanged. V(M) is then incremented by one and Vsc is set to false.

When an S-SYNC-MINOR indication is received, the associated synchronization point serial number, which is indicated to the SS-user, is equal to V(M). V(R) remains unchanged. V(A) is set to V(M) if Vsc is false, otherwise it remains unchanged. V(M) is then incremented by one and Vsc is set to true.

When an S-SYNC-MINOR response is issued, Vsc is true and the associated synchronization point serial number, which is supplied by the SS-user, must be less than V(M) and equal to or greater than V(A). V(A) is set to the serial number plus one. V(M), V(R) and Vsc remain unchanged.

When an S-SYNC-MINOR confirm is received, Vsc is false and the associated synchronization point serial number, which is indicated to the SS-user, is less than V(M) and equal to or greater than V(A). V(A) is set to the serial number plus one. V(M), V(R) and Vsc remain unchanged.

11.4.4 *Major synchronization point*

When an S-SYNC-MAJOR request is issued, the associated synchronization point serial number, which is indicated to the SS-user, is equal to V(M). V(R) remains unchanged. V(A) is set to V(M) if Vsc is true, otherwise it remains unchanged. V(M) is then incremented by one and Vsc is set to false.

When an S-SYNC-MAJOR indication is received, the associated synchronization point serial number, which is indicated to the SS-user, is equal to V(M). V(R) and Vsc remain unchanged. V(A) is set to V(M) if Vsc is false, otherwise it remains unchanged. V(M) is then incremented by one.

When an S-SYNC-MAJOR response is issued, the associated synchronization point serial number is equal to V(M) minus one. No synchronization point serial number is passed with this primitive. V(A) and V(R) are set to V(M). V(M) and Vsc remain unchanged.

When an S-SYNC-MAJOR confirm is received, the associated synchronization point serial number is equal to V(M) minus one. No synchronization point serial number is passed with this primitive. V(A) and V(R) are set to V(M). V(M) and Vsc remain unchanged.

11.4.5 *Resynchronization*

When an S-RESYNCHRONIZE request is issued:

a) if the option is "abandon", there is no associated synchronization point serial number;

b) if the option is "restart", the associated synchronization point serial number, which is supplied by the SS-user, must be greater than or equal to V(R) and less than or equal to V(M);

c) if the option is "set", the associated synchronization point serial number, which is supplied by the SS-user, may have any valid value.

For all options, V(A), V(M), V(R) and Vsc remain unchanged.

When an S-RESYNCHRONIZE indication is received:

d) if the option is "abandon", the associated synchronization point serial number, which is indicated to the SS-user, is greater than or equal to V(M). V(M) is set to the serial number contained in the indication;

e) if the option is "restart", the associated synchronization point serial number, which is indicated to the SS-user, is greater than or equal to V(R). If the synchronization point serial number is greater than V(M) (see Note), the SS-user either responds to the S-RESYNCHRONIZE indication (see g)) or generates a collision (see § 16);

Note – This situation can arise if the extended control QOS is provided and the S-RESYNCHRONIZE request caused an earlier S-SYNC-MINOR request to be discarded by the SS-provider.

f) if the option is "set", the associated synchronization point serial number, which is indicated to the SS-user, may have any valid value.

For all options, V(A), V(R) and Vsc remain unchanged. For the "restart" and "set" options, V(M) remains unchanged.

When an S-RESYNCHRONIZE response is issued:

g) if the option is "abandon", or "restart", the associated synchronization point serial number, which is supplied by the SS-user, must be equal to the value received in the S-RESYNCHRONIZE indication;

h) if the option is "set", the associated synchronization point serial number, which is supplied by the SS-user, may have any valid value.

V(A) and V(M) are set to the synchronization point serial number and Vsc remains unchanged. V(R) is set to zero for the options "abandon" and "set"; it remains unchanged for the "restart" option.

When an S-RESYNCHRONIZE confirm is received:

i) if the option is "abandon", the associated synchronization point serial number, which is indicated to the SS-user, is greater than or equal to V(M);

j) if the option is "restart", the associated synchronization point serial number, which is indicated to the SS-user, is equal to the synchronization point serial number in the corresponding request;

k) if the option is "set", the associated synchronization point serial number, which is indicated to the SS-user, may have any valid value.

V(A) and V(M) are set to the synchronization point serial number and Vsc remains unchanged. V(R) is set to zero for the options "abandon" and "set"; it remains unchanged for the "restart" option.

11.4.6 *Activity management*

When an S-ACTIVITY-START request is issued, or when an S-ACTIVITY-START indication is received, V(A), V(M) and V(R) are set to one and Vsc remains unchanged.

When an S-ACTIVITY-RESUME request is issued, or when an S-ACTIVITY-RESUME indication is received, V(A) and V(M) are set to the synchronization point serial number supplied by the SS-user plus one; V(R) is set to one and Vsc remains unchanged.

The management of V(A), V(M), V(R) and Vsc for S-ACTIVITY-END request, indication, response and confirm is identical to that for S-SYNC-MAJOR request, indication, response and confirm respectively.

The use of S-ACTIVITY-DISCARD and S-ACTIVITY-INTERRUPT primitives has no implication on V(A), V(M), V(R) and Vsc.

12 Session connection establishment phase

12.1 *Session connection service*

12.1.1 *Function*

The session connection service enables two SS-users to establish a session connection between themselves.

Simultaneous attempts by both SS-users to establish a session connection between themselves may result in two session connections. An SS-user may always reject an unwanted connection. No architectural restrictions are placed on the number of concurrent session connections between two SS-users.

This service allows the SS-users to exchange the values of session connection parameters. By the end of the session connection establishment phase, the SS-users have agreed on a set of parameter values concerning the session connection.

12.1.2 *Types of primitives and their parameters*

Table 9/X.215 specifies the types of session service primitives and parameters needed for session connection establishment.

12.1.2.1 *Session Connection Identifier* is a parameter which is provided by the SS-users to enable them to identify the session connection. The Session Connection Identifier is transparent to the SS-provider. This parameter consists of:

a) Calling SS-user Reference (request and indication only) with a maximum of 64 octets;

b) Called SS-user Reference (response and confirm only) with a maximum of 64 octets;

c) Common Reference with a maximum of 64 octets;

d) Additional Reference Information with a maximum of 4 octets.

12.1.2.2 *Calling Session Address* is the session address of the calling entity (see ISO 7498-3).

12.1.2.3 *Calling Session Address* is the session address of the called entity (see ISO 7498-3).

12.1.2.4 *Responding Session Address* is the session address of the responding entity (see ISO 7498-3).

TABLE 9/X.215

Session Connection Establishment Primitives and Parameters

Parameter \ Primitive	S-CONNECT			
	Request	Indication	Response	Confirm
Session connection identifier	U	C(=)	U	C(=)
Calling session address	M	M		
Called session address	M	M		
Responding session address			M	M
Result			M	M(=)
Quality of service	M	M	M	M
Session requirements	M	M(=)	M	M(=)
Initial synchronization point serial number	C	C(=)	C	C(=)
Initial assignment of tokens	C	C(=)	C	C(=)
SS-user data	U	C(=)	U	C(=)

M: presence of the parameter is mandatory.
C: presence of the parameter is conditional.
U: presence of the parameter is a user option.
Blank: the parameter is absent.
(=): the value of the parameter is identical to the value of the corresponding parameter of the preceding SS primitive.

12.1.2.5 *Result* is a parameter indicating the success or failure of the connection establishment request. Its value can be one of:

a) accept;

b) reject by called SS-user, where the reason for failure in the result parameter is one of:
   1) reason not specified;
   2) rejected by called SS-user due to temporary congestion;
   3) rejection by called SS-user. The user data field may be used to provide further information;

c) reject by SS-provider where the reason of failure in the result parameter is one of:
   1) reason not specified;
   2) SS-provider congestion;
   3) called session address unknown;
   4) called SS-user not attached to SSAP;

   Reasons 3) and 4) may be regarded as persistent.

   Only value a) or b) can be present in a response. Any of the values may be present in a confirm.

   5) Implementation restriction state in the PICS.

12.1.2.6 *Quality of service* is a list of parameters which are defined and negotiated as described in § 10.

12.1.2.7 *Session Requirements* is a list of functional units subject to the restrictions defined in § 9.2 and are chosen from:

a) half-duplex functional unit;
b) duplex functional unit;
c) exceptions functional unit;
d) typed data functional unit;
e) negotiated release functional unit;
f) minor synchronize functional unit;
g) major synchronize functional unit;
h) resynchronize functional unit;
i) expedited data functional unit;
j) activity management functional unit;
k) capability data exchange functional unit.

The session requirements specified in the response indicate the called SS-user session requirements to the requestor. The acceptor may not propose both the half-duplex and the duplex functional units in the response. If only one of the half-duplex or duplex functional units was proposed in the indication, then the acceptor proposes the same functional unit in the response or refuses the connection. If the capability data exchange functional unit is proposed, the activity management functional unit is also proposed. If the exceptions functional unit is proposed, the half-duplex functional unit is also proposed. With these exceptions, additional SS-user session requirements which were not included in the indication, may be included in the response. SS-user session requirements that are proposed in both the indication and the response are the ones selected for use on the session connection.

12.1.2.8 *Initial Synchronization Point Serial Number* identifies the initial synchronization point. The conditions for its presence and rules for its negotiation are defined in § 7.6.3. Its value is in the range 0 to 999999.

12.1.2.9 *Initial Assignment of Tokens* is a list of the initial sides to which the available tokens are assigned. The parameter is only required if the corresponding tokens are available. For each available token, the value in a request/indication may be one of:

a) requestor side;
b) acceptor side;
c) acceptor chooses.

The parameter in a response/confirm is absent, unless the value in the request/indication is c), in which case the acceptor replies with a) or b).

12.1.2.10 *SS-user data* is a parameter containing an unlimited number of octets of user information.

12.1.3 *Sequence of primitives*

The sequence of primitives for session connection establishment, whether accepted or rejected, is defined by the time sequence diagram shown in Figure 5/X.215.

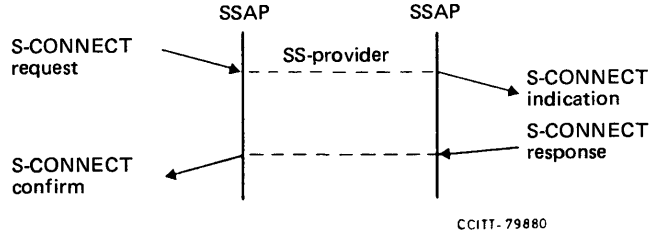


FIGURE 5/X.215

13 Data transfer phase

13.1 *Normal data transfer service*

13.1.1 *Function*

The normal data transfer service allows both SS-users to transfer NSSDUs over the session connection. The SS-provider should deliver each NSSDU to the SS-user as soon as possible. This service is always available on every session connection.

Use of this service is subject to the token restrictions specified in Table 8/X.215.

13.1.2 *Types of primitives and their parameters*

Table 10/X.215 specifies the types of session service primitives and parameters needed for normal data transfer.

TABLE 10/X.215

Normal data transfer primitives and parameters

Parameter \ Primitive	S-Data	
	Request	Indication
SS-user data	M	M(=)

M: presence of the parameter is mandatory.
(=): the value of the parameter is identical to the value of the corresponding parameter of the preceding SS primitive.

SS-user data parameter is an NSSDU. The size of an NSSDU is an integral number of octets greater than zero and unlimited in length.

13.1.3 *Sequence of primitives*

The sequence of primitives in a successful normal data transfer is defined by the time sequence diagram shown in Figure 6/X.215.

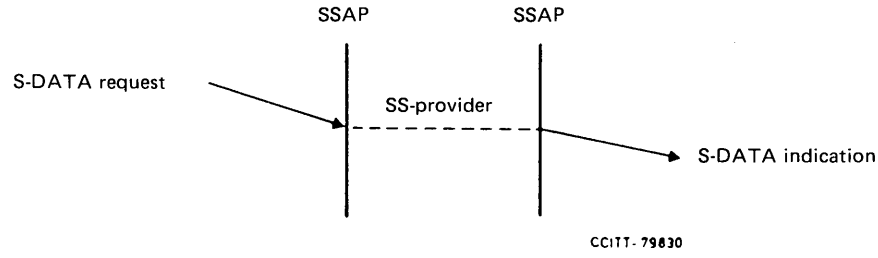


FIGURE 6/X.215

13.2 *Expedited data transfer service*

13.2.1 *Function*

The expedited data transfer service allows SS-users to transfer XSSDUs over the session connection. The transfer of an XSSDU is free from the token and flow control constraints of the normal data transfer service, typed data transfer service and the capability data exchange service.

The SS-provider guarantees that an XSSDU will not be delivered after any subsequently submitted NSSDU or TSSDU on that session connection. The size of an XSSDU is limited.

13.2.2 *Types of primitives and their parameters*

Table 11/X.215 specifies the types of session service primitives and parameters needed for expedited data transfer.

TABLE 11/X.215

Expedited data transfer primitives and parameters

Parameter \ Primitive	S-Expedited-Data	
	Request	Indication
SS-user data	M	M(=)

M: presence of the parameter is mandatory.
(=): the value of the parameter is identical to the value of the corresponding parameter of the preceding SS primitive.

SS-user data parameter is an XSSDU. The size of an XSSDU is 1 to 14 octets.

13.2.3 *Sequence of primitives*

The sequence of primitives in a successful expedited data transfer is defined by the time sequence diagram shown in Figure 7/X.215.

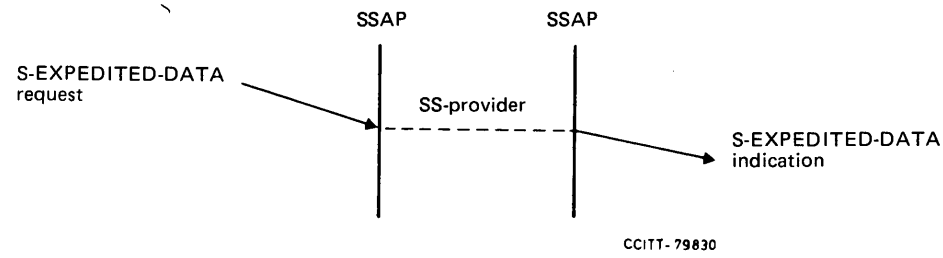


FIGURE 7/X.215

13.3 *Typed data transfer service*

13.3.1 *Function*

The typed data transfer service permits the SS-users to transfer TSSDUs over the session connection. Typed data transfers are subject to the same service restrictions as normal data transfers, except that typed data transfers are not subject to token restrictions.

13.3.2 *Types of primitives and their parameters*

Table 12/X.215 specifies the types of session service primitives and parameters needed for the typed data transfer service.

TABLE 12/X.215

Typed data primitives and parameters

Parameter \ Primitive	S-Typed-Data	
	Request	Indication
SS-user data	M	M(=)

M: presence of the parameter is mandatory.
(=): the value of the parameter is identical to the value of the corresponding parameter of the preceding SS primitive.

SS-user data parameter is a TSSDU. The size of a TSSDU is an integral number of octets greater than zero and unlimited in length.

13.3.3 *Sequence of primitives*

The sequence of primitives in a successful typed data transfer is defined by the time sequence diagram shown in Figure 8/X.215.

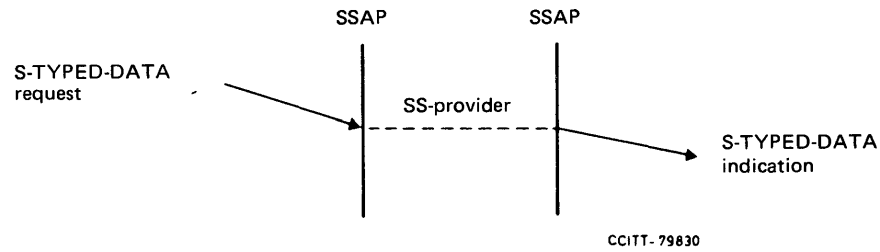


FIGURE 8/X.215

13.4 *Capability data exchange service*

13.4.1 *Function*

The capability data exchange service allows SS-users to exchange user data while not within an activity. The service can only be initiated if activity services are available but no activity is in progress. Use of this service is subject to the token restrictions specified in Table 8/X.215.

13.4.2 *Types of primitives and their parameters*

Table 13/X.215 specifies the types of session service primitives and parameters needed for the capability data exchange service.

TABLE 13/X.215

Capability data exchange primitives and parameters

Parameter \ Primitive	S-Capability-Data			
	Request	Indication	Response	Confirm
SS-user data	U	C(=)	U	C(=)

C: presence of the parameter is conditional.
U: presence of the parameter is a user option.
(=): the value of the parameter is identical to the value of the corresponding parameter of the preceding SS primitive.

SS-user data is a parameter containing an unlimited number of octets of user information.

13.4.3 *Sequence of primitives*

The sequence of primitives in a successful capability data exchange is defined by the time sequence diagram shown in Figure 9/X.215.

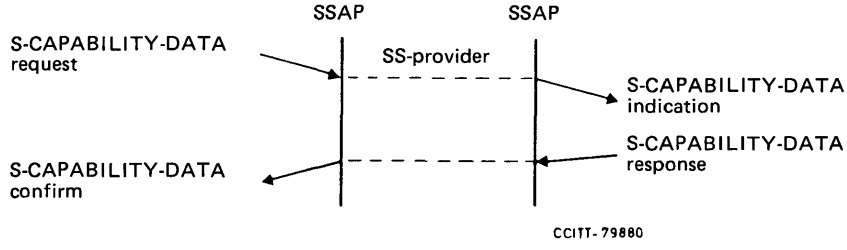


FIGURE 9/X.215

13.5 *Give tokens service*

13.5.1 *Function*

The give tokens service allows an SS-user to surrender one or more tokens to the other SS-user, subject to the token restrictions specified in Table 8/X.215.

The initial assignment of the tokens is established when the session connection is established (see § 7.6.2).

13.5.2 *Types of primitives and their parameters*

Table 14/X.215 specifies the types of session service primitives and parameters needed for the give tokens service.

TABLE 14/X.215

Give tokens primitives and parameters

Parameter \ Primitive	S-Token-Give	
	Request	Indication
Tokens	M	M(=)
SS-user data	U	C(=)

M: presence of the parameter is mandatory.
U: presence of the parameter is conditional.
C: presence of the parameter is a user option.
(=): the value of the parameter is identical to the value of the corresponding parameter of the preceding SS primitive.

13.5.2.1 *Tokens* is a list of tokens assigned to this SS-user to be transferred to the other user. The value is any combination of:

a) data token;
b) synchronize-minor token;
c) major/activity token;
d) release token.

13.5.2.2 *SS-user data* is a parameter containing an unlimited number of user information.

13.5.3 *Sequence of primitives*

The sequence of primitives in a successful transfer of tokens is defined by the time sequence diagram shown in Figure 10/X.215.

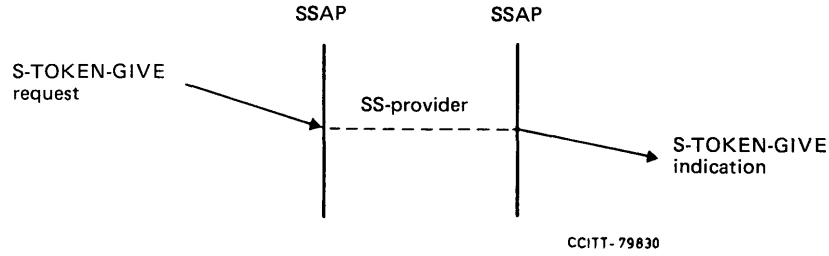


FIGURE 10/X.215

13.6 *Please tokens service*

13.6.1 *Function*

The please tokens service allows an SS-user to request specific tokens, subject to the token restrictions specified in Table 8/X.215.

13.6.2 *Types of primitives and their parameters*

Table 15/X.215 specifies the types of session service primitives and parameters needed for the please tokens service.

TABLE 15/X.215

Please tokens primitives and parameters

Parameter \ Primitive	S-Token-Please	
	Request	Indication
Tokens	M	M(=)
SS-user data	U	C(=)

M: presence of the parameter is mandatory.
C: presence of the parameter is conditional.
U: presence of the parameter is a user option.
(=): the value of the parameter is identical to the value of the corresponding parameter of the preceding SS primitive.

13.6.2.1 *Tokens* is a list of available tokens not assigned to but requested by the SS-user. The value is any combination of:

a) data token;
b) synchronize-minor token;
c) major/activity token;
d) release token.

13.6.2.2 *SS-user data* is a parameter containing an unlimited number of octets of user information.

13.6.3 *Sequence of primitives*

The sequence of primitives in a successful request for tokens is defined by the time sequence diagram shown in Figure 11/X.215.

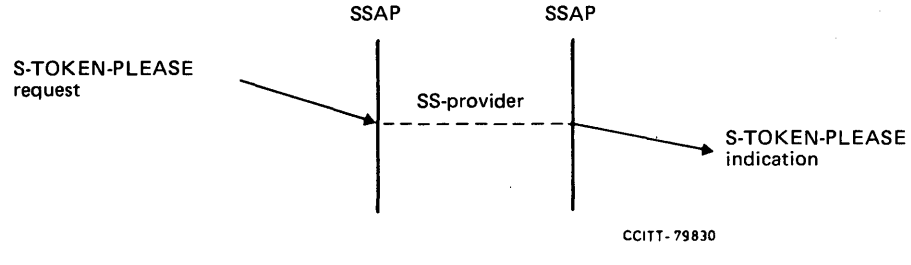


FIGURE 11/X.215

13.7 *Give control service*

13.7.1 *Function*

The give control service allows an SS-user to surrender the entire set of available tokens. This service is an integral part of the activity management concept. This service can only be requested when activity management functional unit has been selected, but no activity is in progress.

13.7.2 *Types of primitives and their parameters*

Table 16/X.215 specifies the types of session service primitives and parameter needed for the give control service.

TABLE 16/X.215

Give Control Primitives and Parameters

Parameter \ Primitive	S-Control-Give	
	Request	Indication
SS-user data	U	C(=)

C: presence of the parameter is conditional.
U: presence of the parameter is a user option.
(=): the value of the parameter is identical to the value of the corresponding parameter of the preceding SS primitive.

SS-user data is a parameter containing an unlimited number of octets of user information.

13.7.3 *Sequence of primitives*

The sequence of primitives in a successful transfer of tokens is defined by the time sequence diagram shown in Figure 12/X.215.

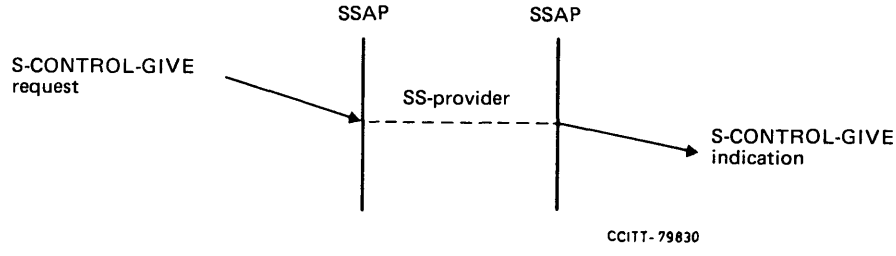


FIGURE 12/X.215

13.8 *Minor synchronization point service*

13.8.1 *Function*

The minor synchronization point service allows SS-users to define minor synchronization points in the flow of NSSDUs and TSSDUs. If the activity management functional unit has been selected, this service can only be initiated within an activity. Use of this service is subject to the token restrictions specified in Table 8/X.215.

The requestor may request explicit confirmation of a minor synchronization point request through the use of the Type parameter. However, the SS-provider does not require that an explicit confirmation be issued. The acceptor may issue a confirmation even if explicit confirmation is not requested.

Responses are issued in the order in which the corresponding indications were received. A further minor synchronization point request may be made while previous minor synchronization points are unconfirmed.

The confirmation of a minor or major synchronization point confirms all previously unconfirmed minor synchronization points. The number of unconfirmed minor synchronization points is not limited by the SS-provider.

Any semantics associated with request and confirmation of a minor synchronization point have no connotations to the SS-provider.

Note – When the duplex functional unit is selected, additional arrangements between SS-users may be required to correlate minor synchronization point requests and confirms with the flow of data from the SS-user without the synchronize-minor token.

13.8.2 *Types of primitives and their parameters*

Table 17/X.215 specifies the types of session service primitives and parameters needed for the minor synchronization point service.

TABLE 17/X.215

Minor synchronization point primitives and parameters

Parameter \ Primitive	S-Sync-Minor			
	Request	Indication	Response	Confirm
Type	M	M(=)		
Synchronization Point Serial Number	M	M(=)	M	M(=)
SS-user data	U	C(=)	U	C(=)

M: presence of the parameter is mandatory.
C: presence of the parameter is conditional.
U: presence of the parameter is a user option.
Blank: the parameter is absent.
(=): the value of the parameter is identical to the value of the corresponding parameter of the preceding SS primitive.

13.8.2.1 *Type* is a parameter which indicates whether or not explicit confirmation is requested by the SS-user and is transparent to the SS-provider. Its value is one of:

a) explicit;

b) optional.

13.8.2.2 *Synchronization Point Serial Number* is defined in § 11.4.3. It is in the range 0 to 999998.

13.8.2.3 *SS-user data* is a parameter containing an unlimited number of octets of user information.

13.8.3 *Sequence of primitives*

The sequence of primitives for confirmation of a minor synchronization point is defined by the time sequence diagram shown in Figure 13/X.215.

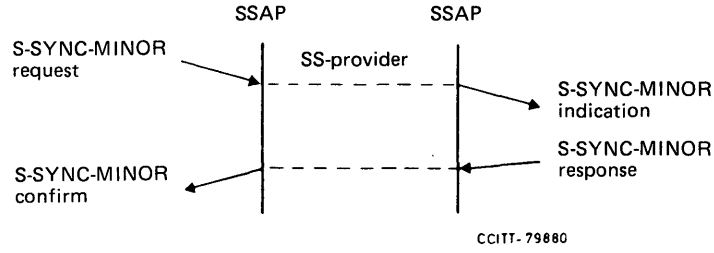


FIGURE 13/X.215

The response and confirm may be absent even if the Type parameter is set to explicit in the indication.

The successful confirmation of the minor synchronization point may also be achieved by issuing (instead of the S-SYNC-MINOR response to the synchronization point specified in the S-SYNC-MINOR indication):

a) an S-SYNC-MINOR response to a subsequent S-SYNC-MINOR indication;

b) an S-SYNC-MAJOR response to a subsequent S-SYNC-MAJOR indication;

c) an S-SYNC-MINOR request for a subsequent minor synchronization point (provided that the synchronize-minor token has been passed from the other SS-user);

d) an S-SYNC-MAJOR request for a subsequent major synchronization point (provided that the synchronize-minor token and, if necessary, the major/activity token have been passed from the other SS-user).

13.9 *Major synchronization point service*

13.9.1 *Function*

The major synchronization point service allows the requestor to define major synchronization points in the flow of NSSDUs, TSSDUs and XSSDUs, to completely separate the flow before and after the major synchronization point. If the activity management functional unit has been selected, this service may only be initiated within an activity. Use of this service is subject to the token restrictions specified in Table 8/X.215.

After making the S-SYNC-MAJOR request, the requestor is not able to initiate any services, except for S-TOKEN-GIVE request, S-ACTIVITY-INTERRUPT request, S-ACTIVITY-DISCARD request, S-U-ABORT request or S-RESYNCHRONIZE request until the S-SYNC-MAJOR confirm is received.

After receiving the S-SYNC-MAJOR indication, in addition to any existing restrictions, the acceptor is not able to initiate S-SYNC-MAJOR request, S-SYNC-MINOR request, S-ACTIVITY-INTERRUPT request, S-ACTIVITY-DISCARD request, S-ACTIVITY-END request or S-RELEASE request until an S-SYNC-MAJOR response is issued.

Expedited data transfer services initiated by the acceptor after issuing a S-SYNC-MAJOR response are not indicated before the S-SYNC-MAJOR confirm.

13.9.2 *Types of primitives and their parameters*

Table 18/X.215 specifies the types of session service primitives and parameters needed for the major synchronization point service.

TABLE 18/X.215

Major synchronization point primitives and parameters

Parameter \ Primitive	S-Sync-Major			
	Request	Indication	Response	Confirm
Synchronization Point Serial Number	M	M(=)		
SS-user data	U	C(=)	U	C(=)

M: presence of the parameter is mandatory.
C: presence of the parameter is conditional.
U: presence of the parameter is a user option.
Blank: the parameter is absent.
(=): the value of the parameter is identical to the value of the corresponding parameter of the preceding SS primitive.

13.9.2.1 *Synchronization Point Serial Number* is defined in § 11.4.4. It is in the range 0 to 999998.

13.9.2.2 *SS-user data* is a parameter containing an unlimited number of octets of user information.

13.9.3 *Sequence of primitives*

The sequence of primitives in the successful definition of a major synchronization point is defined by the time sequence diagram shown in Figure 14/X.215.

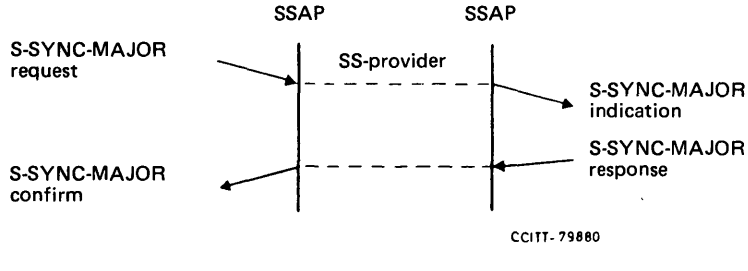


FIGURE 14/X.215

13.10 *Resynchronize service*

13.10.1 *Function*

The resynchronize service is provided to assist orderly reestablishment of communication within the current session connection, typically following an error or lack of response by either of the SS-user or the SS-provider, or disagreements between SS-users. Requesting the service sets the session connection to an agreed defined state, including the positions of the available tokens and the value of the synchronization point serial number, which will be the next synchronization point serial number to be used.

The service may be initiated by either SS-user and has the following characteristics:

a) after issuing the S-RESYNCHRONIZE request, the requestor is not able to initiate any services except S-U-ABORT request, until the S-RESYNCHRONIZE confirm is received;

b) after having received an S-RESYNCHRONIZE indication, the acceptor may only issue:

   1) S-RESYNCHRONIZE response; or

   2) S-RESYNCHRONIZE request (see note); or

   3) S-ACTIVITY-DISCARD request (see note); or

   4) S-ACTIVITY-INTERRUPT request (see note); or

   5) S-U-ABORT request.

   Note – These requests cause a collision of resynchronize requests and therefore the SS-user can only issue the request if he is going to be the collision winner (see § 16).

c) all undelivered data are purged;

d) means are provided for the requesting SS-user either to set or to let the acceptor set a new assignment of each available token;

e) means are provided to assign a new value for the synchronization point serial number;

f) when there is an unacknowledged major synchronization point at the time of the S-RESYNCHRONIZE indication, this point remains unacknowledged. In any case, no confirmations should be issued until the resynchronization is complete and until new indications for synchronization points have been received;

g) collision of resynchronize requests is resolved, so that only one of the colliding requests is confirmed (see § 16).

The Resynchronize Type parameter is used to indicate the resynchronize option:

h) *abandon* is used to request the SS-provider to resynchronize the session connection to a new synchronization point which is greater than or equal to V(M). The new synchronization point serial number will be greater than any previous value used on this session connection. Where there are unacknowledged minor synchronization points at the time of the S-RESYNCHRONIZE request/indication, they remain unacknowledged;

i) *restart* is used to return to an agreed point which is identified by a past acknowledged synchronization point serial number. This point cannot be earlier than the last confirmed major synchronization point. The necessary securing of state information associated with the point is the responsability of the SS-users;

j) *set* is used to synchronize to any valid synchronization point serial number specified by the SS-users. When there are unacknowledged minor synchronization points at the time of the S-RESYNCHRONIZE request/indication, they remain unacknowledged.

13.10.2 *Types of primitives and their parameters*

Table 19/X.215 specifies the types of session service primitives and parameters needed for the resynchronize service.

TABLE 19/X.215

Resynchronize primitives and parameters

Parameter \ Primitive	S-Resynchronize			
	Request	Indication	Response	Confirm
Resynchronize Type	M	M(=)		
Synchronization Point Serial Number	C	M	M	M(=)
Assignment of Tokens	C	C(=)	C	C(=)
SS-user data	U	C(=)	U	C(=)

M: presence of the parameter is mandatory.
C: presence of the parameter is conditional.
U: presence of the parameter is a user option.
Blank: the parameter is absent.
(=): the value of the parameter is identical to the value of the corresponding parameter of the preceding SS primitive.

13.10.2.1 *Resynchronize Type* is a parameter which specifies one of the resynchronize options. Its value is one of:

a) abandon;

b) restart;

c) set.

13.10.2.2 *Synchronization Point Serial Number* depends on the resynchronize option and is defined in §§ 11.4 and 11.4.5.

13.10.2.3 Assignment of *Tokens* is a list of the available tokens for the session connection with values for their assignment following the resynchronization. For each available token, the value in a request/indication is one of:

a) requestor side;

b) acceptor side;

c) acceptor chooses.

The value for a response/confirm is the same as in the request/indication unless that value is c), in which case the acceptor chooses a) or b).

13.10.2.4 *SS-user data* is a parameter containing an unlimited octets of user information.

13.10.3 *Sequence of primitives*

The sequence of primitives in a successful resynchronization without collision is defined by the time sequence diagram shown in Figure 15/X.215. Collision cases are defined in § 16.

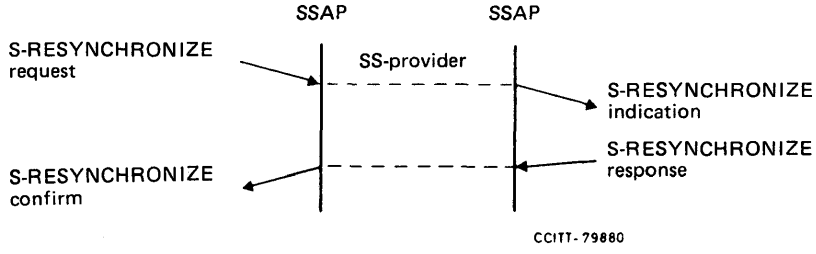


FIGURE 15/X.215

13.11 *P-exception reporting service*

13.11.1 *Function*

The P-exception reporting service permits SS-users to be notified of unanticipated situations not covered by other services. If a service cannot be completed due to SS-provider protocol errors or malfunctions, the P-exception reporting service is used to indicate this to both SS-users.

If used with the activity management service, the P-exception reporting service is only permitted while an activity is in progress or waiting for S-CAPABILITY-DATA confirm.

Following an S-P-EXCEPTION-REPORT indication, and until the error condition is cleared:

a) NSSDUs, TSSDUs and XSSDUs will be discarded by the SS-provider;

b) synchronization point indications will not be given to the SS-users.

On receipt of an S-P-EXCEPTION-REPORT indication, either SS-user initiates one of the following services to clear the error:

c) resynchronize;

d) abort;

e) activity interrupt or activity discard;

f) give the data token (see notes).

The SS-users are not permitted to initiate any other services until the error is cleared.

Note 1 – It is not recommended that the error condition be cleared by passing the data token when the resynchronize and/or activity management functional units have been selected.

Note 2 – If the error condition is cleared by passing the data token, data and synchronization point serial numbers may be lost. However, the SS-provider will keep track of the serial numbers of the synchronization points which have been discarded. Therefore, the synchronization point serial number indicated to the SS-user in a synchronization point request/indication made after the error condition has been cleared will reflect the fact that synchronization points have been discarded during the error condition.

Note 3 – XSSDUs sent after the S-TOKEN-GIVE request will be discarded if they overtake the request.

Note 4 – Tokens other than the data token may be transferred at the same time.

13.11.2 *Types of primitives and their parameters*

Table 20/X.215 specifies the types of session service primitives and parameters needed for the P-exception reporting service.

TABLE 20/X.215

P-exception reporting primitives and parameters

Primitive / Parameter	S-P-Exception-Report
	Indication
Reason	M

M: presence of the parameter is mandatory.

Reason is a parameter specifying the reason for the exception report. Its value is one of:

a) protocol error;

b) non-specific error.

13.11.3 *Sequence of primitives*

The sequence of primitives in a successful P-exception report is defined by the time sequence diagram shown in Figure 16/X.215.

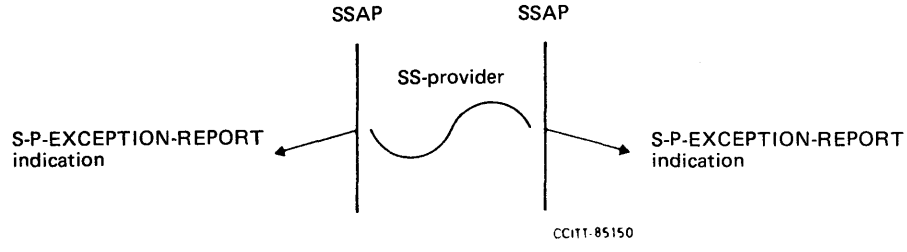


FIGURE 16/X.215

13.12 *U-exception reporting service*

13.12.1 *Function*

The U-exception reporting service permits an SS-user to report an exception condition subject to the token restrictions specified in Table 8/X.215.

If used with the activity management service, the U-exception reporting service is only permitted while an activity is in progress.

Following an S-U-EXCEPTION-REPORT request, and until the error condition is cleared:

a) NSSDUs, TSSDUs and XSSDUs will be discarded by the SS-provider;

b) synchronization point indications will not be given to the requestor of the S-U-EXCEPTION-REPORT;

c) the requestor is only permitted to issue S-U-ABORT request.

On receipt of an S-U-EXCEPTION-REPORT indication, the acceptor initiates one of the following services to clear the error:

d) resynchronize;

e) abort;

f) activity interrupt or activity discard;

g) give the data token (see notes).

The acceptor is not permitted to initiate any other services until the error is cleared.

Note 1 – It is not recommended that the error condition be cleared by passing the data token when the resynchronize and/or activity management functional units have been selected.

Note 2 – If the error condition is cleared by passing the data token, data and synchronization point serial numbers may be lost. However, the SS-provider will keep track of the serial numbers of the synchronization points which have been discarded. Therefore, the synchronization point serial number indicated to the SS-user in a synchronization point request/indication made after the error condition has been cleared will reflect the fact that synchronization points have been discarded during the error condition.

Note 3 – XSSDUs sent after the S-TOKEN-GIVE request will be discarded if they overtake the request.

Note 4 – Tokens other than the data token may be transferred at the same time.

13.12.2 *Types of primitives and their parameters*

Table 21/X.215 specifies the types of session service primitives and parameters needed for the U-exception reporting service.

TABLE 21/X.215

U-exception reporting primitives and parameters

Parameter \ Primitive	S-U-Exception-Report	
	Request	Indication
Reason	M	M(=)
SS-user data	U	C(=)

M: presence of the parameter is mandatory.
C: presence of the parameter is conditional.
U: presence of the parameter is a user option.
(=): the value of the parameter is identical to the value of the corresponding parameter of the preceding SS primitive.

13.12.2.1 *Reason* is a parameter specifying the reason for the exception report and is transparent to the SS-provider. Its value is one of:

a) SS-user receiving ability jeopardized (i.e. data received may not be handled correctly);
b) local SS-user error;
c) sequence error;
d) demand data token;
e) unrecoverable procedural error;
f) non-specific error.

13.12.2.2 *SS-user data* is a parameter containing an unlimited number of octets of user information.

13.12.3 *Sequence of primitives*

The sequence of primitives in a successful U-exception report is defined by the time sequence diagram shown in Figure 17/X.215.

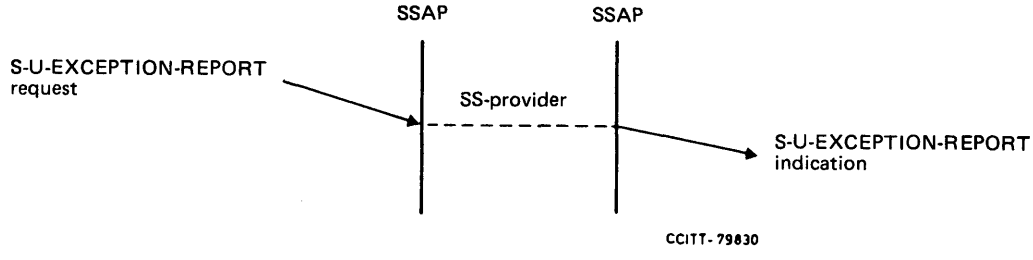


FIGURE 17/X.215

13.13 *Activity start service*

13.13.1 *Function*

The activity start service allows an SS-user to indicate that a new activity is entered. The value of the next synchronization point serial number to be used is set to one (see § 11.4.6). The service can only be initiated if no activity is in progress and subject to the token restrictions specified in Table 8/X.215.

13.13.2 *Types of primitives and their parameters*

Table 22/X.215 specifies the types of session service primitives and parameters needed for the activity start service.

TABLE 22/X.215

Activity start primitives and parameters

Parameter \ Primitive	S-Activity-Start	
	Request	Indication
Activity Identifier	M	M(=)
SS-user data	U	C(=)

M: presence of the parameter is mandatory.
C: presence of the parameter is conditional.
U: presence of the parameter is a user option.
(=): the value of the parameter is identical to the value of the corresponding parameter of the preceding SS primitive.

13.13.2.1 *Activity identifier* is a parameter which is provided by the SS-users to enable them to identify the new activity and is transparent to the SS-provider. This parameter has a maximum of 6 octets.

13.13.2.2 *SS-user data* is a parameter containing an unlimited number of octets of user information.

13.13.3 *Sequence of primitives*

The sequence of primitives in a successful activity start is defined by the time sequence diagram shown in Figure 18/X.215.

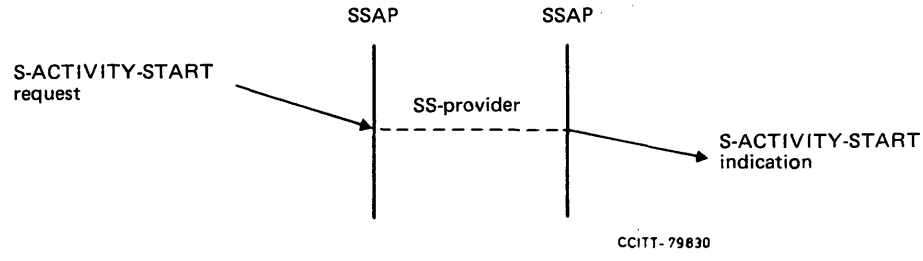


FIGURE 18/X.215

13.14 *Activity resume service*

13.14.1 *Function*

The activity resume service allows an SS-user to indicate that a previously interrupted activity is resumed. A new activity identifier is provided by the SS-user together with the identifier of the activity being resumed and the next synchronization point serial number to be used minus one. In the case when the resumed activity was originally started on another session connection, the session connection identifier of that session connection is also provided by the SS-user.

The service can only be initiated if no activity is in progress and subject to the token restrictions specified in Table 8/X.215.

13.14.2 *Types of primitives and their parameters*

Table 23/X.215 specifies the types of session service primitives and parameters needed for the activity resume.

TABLE 23/X.215

Activity resume primitives and parameters

Parameter \ Primitive	S-Activity-Resume	
	Request	Indication
Activity Identifier	M	M(=)
Old Activity Identifier	M	M(=)
Synchronization Point Serial Number	M	M(=)
Old Session Connection Identifier	U	C(=)
SS-user data	U	C(=)

M: presence of the parameter is mandatory.
C: presence of the parameter is conditional.
U: presence of the parameter is a user option.
(=): the value of the parameter is identical to the value of the corresponding parameter of the preceding SS primitive.

13.14.2.1 *Activity Identifier* is a parameter which is provided by the SS-users to enable them to give a new identifier to the activity being resumed and is transparent to the SS-provider. This parameter has a maximum of 6 octets.

13.14.2.2 *Old Activity Identifier* is the original identifier of the activity being resumed and is transparent to the SS-provider.

13.14.2.3 *Synchronization Point Serial Number* is provided by the SS-user and is defined in § 11.4.6.

13.14.2.4 *Old Session Connection Identifier* is the session connection identifier of the session connection in which the activity being resumed was originally started and is transparent to the SS-provider. It consists of:

a) Calling SS-user Reference with a maximum of 64 octets;
b) Called SS-user Reference with a maximum of 64 octets;
c) Common Reference with a maximum of 64 octets;
d) Additional Reference Information with a maximum of 4 octets.

13.14.2.5 *SS-user data* is a parameter containing an unlimited number of octets of user information.

13.14.3 *Sequence of primitives*

The sequence of primitives in a successful activity resume is defined by the time sequence diagram shown in Figure 19/X.215.

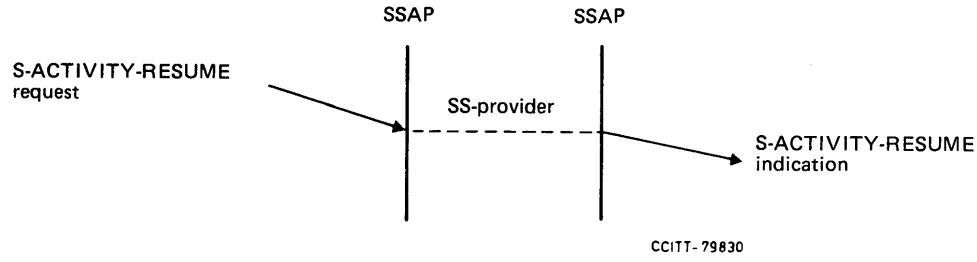


FIGURE 19/X.215

13.15 *Activity interrupt service*

13.15.1 *Function*

The activity interrupt service allows an SS-user to abnormally terminate the current activity so that work achieved before the interruption is not cancelled, and may be resumed later.

The service can only be initiated if an activity is in progress and subject to the token restrictions specified in Table 8/X.215. After receipt of the confirm, all available tokens are assigned to the SS-user which issued the request.

After issuing an S-ACTIVITY-INTERRUPT request, the requestor is not able to initiate any services, except S-U-ABORT request, until the S-ACTIVITY-INTERRUPT confirm is received.

After receiving an S-ACTIVITY-INTERRUPT indication, the acceptor is not able to initiate any services, except S-U-ABORT request, until the S-ACTIVITY-INTERRUPT response is issued.

Use of this service may cause loss of data which has not yet been delivered to the SS-user.

13.15.2 *Types of primitives and their parameters*

Table 24/X.215 specifies the types of session service primitives and parameters needed for the activity interrupt service.

TABLE 24/X.215

Activity interrupt primitives and parameters

Primitive / Parameter	S-Activity-Interrupt			
	Request	Indication	Response	Confirm
Reason	U	C(=)		
SS-user data	U	C(=)	U	C(=)

C: presence of the parameter is conditional.
U: presence of the parameter is a user option.
Blank: the parameter is absent.
(=): the value of the parameter is identical to the value of the corresponding parameter of the preceding SS primitive.

13.15.2.1 *Reason* is a parameter specifying the reason for the activity interrupt and is transparent to the SS-provider. Its value is one of:

a) SS-user receiving ability jeopardized (i.e. data received may not be handled correctly);

b) local SS-user error;

c) sequence error;

d) demand data token;

e) unrecoverable procedural error;

f) non-specific error.

13.15.2.2 *SS-user data* is a parameter containing an unlimited number of octets of user information.

13.15.3 *Sequence of primitives*

The sequence of primitives in a successful activity interrupt is defined by the time sequence diagram shown in Figure 20/X.215.

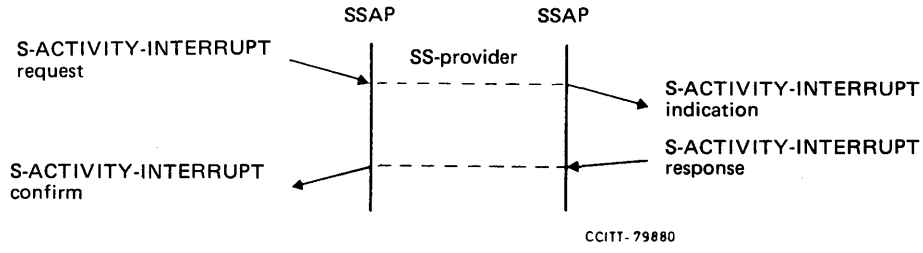


FIGURE 20/X.215

13.16 *Activity discard service*

13.16.1 *Function*

The activity discard service allows an SS-user to abnormally terminate the current activity. There is an implied meaning to the SS-user that the previous content of this activity is cancelled, but this is not controlled by the SS-provider.

The service can only be initiated if an activity is in progress and subject to the token restrictions specified in Table 8/X.215. After receipt of the confirm, all available tokens are assigned to the SS-user which issued the request.

After issuing an S-ACTIVITY-DISCARD request, the requestor is not able to initiate any services, except S-U-ABORT request, until the S-ACTIVITY-DISCARD confirm is received.

After receiving an S-ACTIVITY-DISCARD indication, the acceptor is not able to initiate any services, except S-U-ABORT request, until the S-ACTIVITY-DISCARD response is issued.

Use of this service may cause loss of data which has not yet been delivered to the SS-user.

13.16.2 *Types of primitives and their parameters*

Table 25/X.215 specifies the types of session service primitives and parameters needed for the activity discard service.

TABLE 25/X.215

Activity discard primitives and parameters

Parameter \ Primitive	S-Activity-Discard			
	Request	Indication	Response	Confirm
Reason	U	C(=)		
SS-user data	U	C(=)	U	C(=)

C: presence of the parameter is conditional.
U: presence of the parameter is a user option.
Blank: the parameter is absent.
(=): the value of the parameter is identical to the value of the corresponding parameter of the preceding SS primitive.

13.16.2.1 *Reason* is a parameter specifying the reason for the activity discard and is transparent to the SS-provider. Its value is one of:

a) SS-user receiving ability jeopardized (i.e. data received may not be handled correctly);
b) local SS-user error;
c) sequence error;
d) demand data token;
e) unrecoverable procedural error;
f) non-specific error.

13.16.2.2 *SS-user data* is a parameter containing an unlimited number of octets of user information.

13.16.3 *Sequence of primitives*

The sequence of primitives in a successful activity discard is defined by the time sequence diagram shown in Figure 21/X.215.

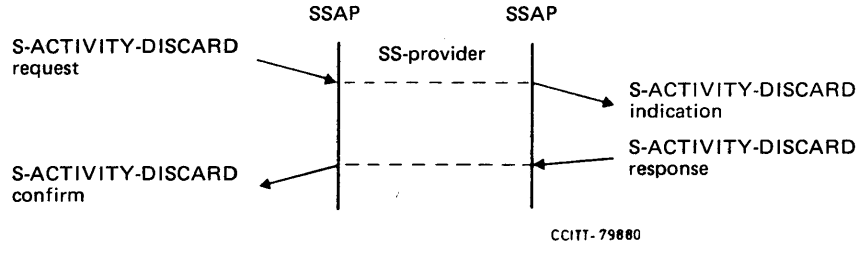


FIGURE 21/X.215

13.17 *Activity end service*

13.17.1 *Function*

The activity end service allows an SS-user to indicate the end of an activity, and has the effect of setting a major synchronization point. This service can only be invoked if an activity is in progress and subject to the token restrictions specified in Table 8/X.215.

After issuing the S-ACTIVITY-END request, in addition to any existing restrictions, the requestor is not able to initiate any services, except for S-U-ABORT request, S-ACTIVITY-INTERRUPT request, S-ACTIVITY-DISCARD request or S-TOKEN-GIVE request until the S-ACTIVITY-END confirm is received.

After receiving the S-ACTIVITY-END indication, in addition to any existing restrictions, the acceptor is not able to initiate S-SYNC-MAJOR request, S-SYNC-MINOR request, S-ACTIVITY-INTERRUPT request, S-ACTIVITY-DISCARD request, S-ACTIVITY-END request or S-RELEASE request until the S-ACTIVITY-END response is issued.

If the activity management functional unit has been selected, the SS-user is not allowed to initiate any services, except activity start, activity resume, token management, capability data, expedited data, typed data, normal data, release or abort, until an activity is started or resumed.

13.17.2 *Types of primitives and their parameters*

Table 26/X.215 specifies the types of session service primitives and parameters needed for the activity end service.

TABLE 26/X.215

Activity end primitives and parameters

Parameter \ Primitive	S-Activity-End			
	Request	Indication	Response	Confirm
Synchronization Point Serial Number	M	M(=)		
SS-user data	U	C(=)	U	C(=)

M: presence of the parameter is mandatory.
C: presence of the parameter is conditional.
U: presence of the parameter is a user option.
Blank: the parameter is absent.
(=): the value of the parameter is identical to the value of the corresponding parameter of the preceding SS primitive.

13.17.2.1 *Synchronization Point Serial Number* is defined in § 11.4.6.

13.17.2.2 *SS-user data* is a parameter containing an unlimited number of octets of user information.

13.17.3 *Sequence of primitives*

The sequence of primitives in a successful normal termination of an activity is defined by the time sequence diagram shown in Figure 22/X.215.

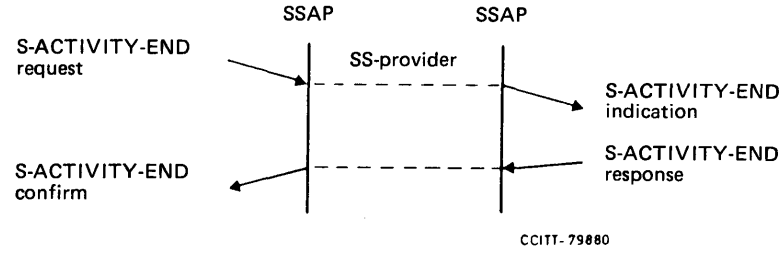


FIGURE 22/X.215

14 Session connection release phase

14.1 *Orderly release service*

14.1.1 *Function*

The orderly release service is always provided and allows either SS-user to release the session connection in an orderly manner. This is done cooperatively between the two SS-users without the loss of data after all in-transit data have been delivered and accepted by both SS-users.

Use of this service is subject to the token restrictions specified in Table 8/X.215. If the release token is available, the acceptor may refuse the release and continue the session connection without loss of data. If the release token is not available, the acceptor cannot refuse the release.

14.1.2 *Types of primitives and their parameters*

Table 27/X.215 specifies the types of session service primitives and parameters needed for the orderly release service.

TABLE 27/X.215

Orderly release primitives and parameters

Parameter \ Primitive	S-Release			
	Request	Indication	Response	Confirm
Result			M	M(=)
SS-user data	U	C(=)	U	C(=)

M: presence of the parameter is mandatory.
C: presence of the parameter is conditional.
U: presence of the parameter is a user option.
Blank: the parameter is absent.
(=): the value of the parameter is identical to the value of the corresponding parameter of the preceding SS primitive.

14.1.2.1 *Result* is a parameter indicating whether or not the session release is granted. Its value may be one of:

a) affirmative;

b) negative.

The latter value may be given only if the release token is available.

14.1.2.2 *SS-user data* is a parameter containing an unlimited number of octets of user information.

14.1.3 *Sequence of primitives*

The sequence of primitives in a successful orderly session release is defined by the time sequence diagram shown in Figure 23/X.215.

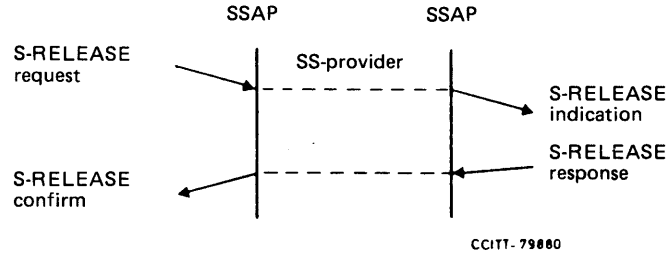


FIGURE 23/X.215

A collision of S-RELEASE requests may occur when no tokens are available. This results in S-RELEASE indications to both SS-user. In this case, the calling SS-user should send the S-RELEASE response after receiving the S-RELEASE indication from the called SS-user. The called SS-user should not send his S-RELEASE response before receiving the S-RELEASE confirm from the calling SS-user.

14.2 *U-abort service*

14.2.1 *Function*

The U-abort service provides the means by which either SS-user can instantaneously release the session connection and have the other SS-user informed of this release. Use of this service will cause loss of undelivered data.

14.2.2 *Types of primitives and their parameters*

Table 28/X.215 specifies the types of session service primitives and parameters needed for the U-abort service.

TABLE 28/X.215

U-abort primitives and parameters

Parameter \ Primitive	S-U-Abort	
	Request	Indication
SS-user data	U	C(=)

C: presence of the parameter is conditional.
U: presence of the parameter is a user option.
(=): the value of the parameter is identical to the value of the corresponding parameter of the preceding SS primitive.

SS-user data is a parameter containing an unlimited number of octets of user information.

14.2.3 *Sequence of primitives*

The sequence of primitives in a successful U-abort is defined by the time sequence diagram shown in Figure 24/X.215.

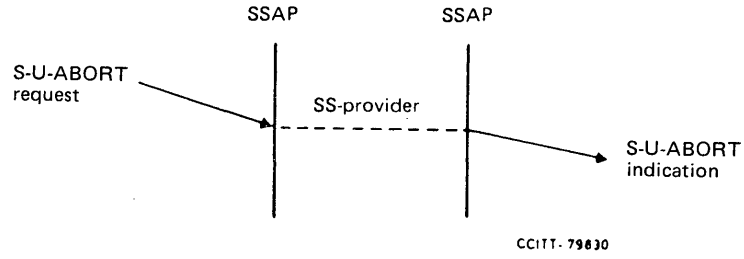


FIGURE 24/X.215

14.3 *P-abort service*

14.3.1 *Function*

The P-abort service provides the means by which the SS-provider may indicate the release of the session connection for reasons internal to the SS-provider. Use of this service will cause loss of undelivered data. A reason code of limited size is passed from the SS-provider to the SS-user.

14.3.2 *Types of primitives and their parameters*

Table 29/X.215 specifies the types of session service primitives and parameters needed for the P-abort service.

TABLE 29/X.215

P-abort primitives and parameters

Primitive / Parameter	S-P-Abort
	Indication
Reason	M

M: presence of the parameter is mandatory.

Reason is a parameter indicating the reason for the abort. Its value is one of:

a) transport disconnect;

b) protocol error;

c) undefined;

d) implementation restriction stated in the PICS.

14.3.3 *Sequence of primitives*

The sequence of primitives in a successful P-abort is defined by the time sequence diagram shown in Figure 25/X.215.

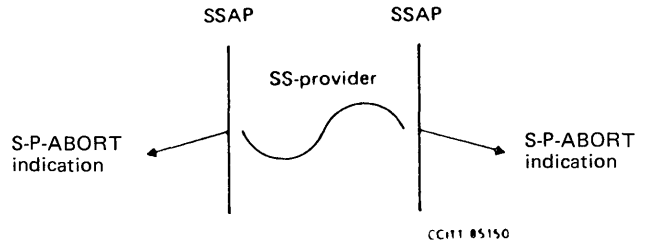


FIGURE 25/X.215

15 Sequences of primitives

15.1 *State tables*

Annex A contains state tables which define the constraints on the sequences in which the session service primitives may occur. The constraints determine the order in which the session services occur, but do not fully specify when they may occur. Other constraints will affect the ability of an SS-user or the SS-provider to issue a primitive at any particular time.

15.2 *Sequences of primitives at one session connection end-point*

The possible sequences of primitives at one session connection end-point may be derived directly from the state tables in Annex A.

16 Collision

16.1 *Collision as viewed by the SS-user*

The SS-provider resolves collisions between those requests that may destroy SS-user data. If a collision occurs, one of the SS-users will receive an unexpected indication while awaiting one of the following:

a) S-RESYNCHRONIZE confirm;

b) S-ACTIVITY-INTERRUPT confirm;

c) S-ACTIVITY-DISCARD confirm;

d) clearing the error state after issuing an S-EXCEPTION-REPORT request.

Table 30/X.215 defines the indications that may be received which indicate that the SS-user has lost a collision resolved by the SS-provider.

TABLE 30/X.215

Indications resulting from collision resolution

SS-user receives / SS-user is waiting for	ER	RR	RS	RA	AI	AD	AB
Clearing error state after S-U-Exception-Report request		X	X	X	X	X	X
S-Resynchronize (restart) confirm		X	X	X	X	X	X
S-Resynchronize (set) confirm			X	X	X	X	X
S-Resynchronize (abandon) confirm				X	X	X	X
S-Activity-Interrupt confirm							X
S-Activity-discard confirm							X

X: indication may be received.
Blank: indication will not be received.
AB: S-P-Abort indication or S-U-Abort indication.
AD: S-Activity-Discard indication.
AI: S-Activity-Interrupt indication.
ER: S-U-Exception-Report indication or S-P-Exception-Report indication.
RA: S-Resynchronize (abandon) indication.
RR: S-Resynchronize (restart) indication.
RS: S-Resynchronize (set) indication.

16.2 *Collision resolution by the SS-provider*

The SS-provider resolves colliding SS-user requests according to the following rules.

In the case of collision between two of the following types of requests, the first in the list takes precedence.

a) S-U-ABORT request;
b) S-ACTIVITY-DISCARD request;
c) S-ACTIVITY-INTERRUPT request;
d) S-RESYNCHRONIZE (abandon) request;
e) S-RESYNCHRONIZE (set) request;
f) S-RESYNCHRONIZE (restart) request;
g) S-U-EXCEPTION-REPORT request.

Possible collisions of the same request are handled as follows:

h) If two S-RESYNCHRONIZE (abandon) requests collide, the calling SS-user request takes precedence.
i) If two S-RESYNCHRONIZE (restart) requests collide, the request with the lowest serial number takes precedence. If the serial numbers are equal, the calling SS-user request takes precedence.
j) If two S-RESYNCHRONIZE (set) requests collide, the calling SS-user request takes precedence.

ANNEX A

(to Recommendation X.215)

State tables

A.1 *General*

This Annex describes the session service in terms of state tables. The state tables show the state of an SS-user, the events that occur at the session service boundary, the actions taken by the SS-user and the resultant state.

These state tables do not constitute a formal definition of the session service; they are included to provide a more precise definition of the relationships between session service primitives defined in §§ 12, 13 and 14.

Table A-1/X.215 specifies the abbreviated name and name of each incoming event generated by the SS-provider.

Table A-2/X.215 specifies the abbreviated name and name of each state.

Table A-3/X.215 specifies the abbreviated name and name of each outgoing event generated by the SS-user.

Table A-4/X.215 summarizes the operations on the variables V(A), V(M), V(R) and Vsc.

Table A-5/X.215 specifies the specific actions.

Table A-6/X.215 specifies the predicates.

Tables A-7/X.215 to 14/X.215 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.215.

A.2.3 Predicates are represented by the notation pn, where n is the number of the predicate in Table A-6/X.215.

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 or outgoing event which is invalid is left blank.

A.3.2 The intersection of each state and incoming or outgoing event which is valid contains entries which are either:

a) an *action list* which:
   1) may contain 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.4 *Actions to be taken by the SS-user*

The state tables define the action to be taken by the SS-user.

A.4.1 *Invalid intersections*

If the intersection of the state and an incoming or outgoing event is invalid, any action taken by the SS-user 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 SS-user 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 SS-user 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 SS-user shall take one of the actions defined in § A.4.1.

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 exchange 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 S-CONNECT response is issued or the S-CONNECT confirm is 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 SS-user;

OWNED(t) = false: if token not assigned to the SS-user.

OWNED(t) is not defined if AV(t) = false.

OWNED(t) is set when:

1) the S-CONNECT response is issued or the S-CONNECT confirm is received; or
2) the S-RESYNCHRONIZE response is issued or the S-RESYNCHRONIZE confirm is received; or
3) the S-TOKEN-GIVE request is issued or the S-TOKEN-GIVE indication is received; or
4) the S-CONTROL-GIVE request is issued or the S-CONTROL-GIVE indication is received;
5) the S-ACTIVITY-INTERRUPT response is issued or the S-ACTIVITY-INTERRUPT confirm is received;
6) the S-ACTIVITY-DISCARD response is issued or the S-ACTIVITY-DISCARD confirm is received.

c) I(t), for t in tk-dom, is a function which, when true, indicates that the SS-user 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 SS-user 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 SS-user 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 SS-user 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 sets 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 S-RELEASE indication)

b) ANY(f, s), for f in F and s in S:

ANY(f, s) = true: any f(t) = true for t in s and 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 *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 S-ACTIVITY-START request or S-ACTIVITY-RESUME request is issued or the S-ACTIVITY-START indication or S-ACTIVITY-RESUME indication is received (only possible when FU(ACT) = true).

c) Vact is set false when the S-ACTIVITY-DISCARD response or S-ACTIVITY-INTERRUPT response is issued or the S-ACTIVITY-DISCARD confirm or S-ACTIVITY-INTERRUPT confirm is received.

d) Vact is set false when the S-ACTIVITY-END response is issued or the S-ACTIVITY-END confirm is received.

A.5.4.2 *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.

Vrspnb indicates the serial number in the case of resynchronize restart.

Vrsp and, if necessary Vrspnb, are set when an S-RESYNCHRONIZE request is issued or an S-RESYNCHRONIZE indication is received. Vrsp is set to no when the SS-user goes to STA713.

A.5.4.3 *Vcoll*

Vcoll is a boolean variable having the following values:

Vcoll = true: a collision of S-RELEASE requests has been detected;

Vcoll = false: there has not been a collision of S-RELEASE requests.

This variable is set false during the session connection establishment phase.

A.5.4.4 *V(A)*

V(A) is used by the SS-user and is the lowest serial number to which a synchronization point confirmation is expected. No confirmation is expected when V(A) = V(M).

A.5.4.5 *V(M)*

V(M) is used by the SS-user and is the next serial number to be used.

A.5.4.6 *V(R)*

V(R) is used by the SS-user and is the lowest serial number to which resynchronization restart is permitted.

A.5.4.7 *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 session connection establishment phase and when an S-SYNC-MINOR request is issued. Vsc is set true when an S-SYNC-MINOR indication is received.

Note – Table A-4/X.215 summarizes the operations on V(A), V(M), V(R) and Vsc.

A.5.4.8 *Vdnr*

Vdnr is a boolean variable having the following values:

Vdnr = true: an S-RELEASE confirm has been received in STA09 (following a collision of S-RELEASE requests);

Vdnr = false: no S-RELEASE confirm has been received.

This variable is set to false during the connection establishment phase.

TABLE A-1/X.215

Events generated by the SS-provider

Abbreviated name	Name and description
SACTDind	S-ACTIVITY-DISCARD indication primitive
SACTDcnf	S-ACTIVITY-DISCARD confirm primitive
SACTEind	S-ACTIVITY-END indication primitive
SACTEcnf	S-ACTIVITY-END confirm primitive
SACTIind	S-ACTIVITY-INTERRUPT indication primitive
SACTIcnf	S-ACTIVITY-INTERRUPT confirm primitive
SACTRind	S-ACTIVITY-RESUME indication primitive
SACTSind	S-ACTIVITY-START indication primitive
SCDind	S-CAPABILITY-DATA indication primitive
SCDcnf	S-CAPABILITY-DATA confirm primitive
SCGind	S-CONTROL-GIVE indication primitive
SCONind	S-CONNECT indication primitive
SCONcnf+	S-CONNECT (accept) confirm primitive
SCONcnf–	S-CONNECT (reject) confirm primitive
SDTind	S-DATA indication primitive
SEXind	S-EXPEDITED-DATA indication primitive
SGTind	S-TOKEN-GIVE indication primitive
SPABind	S-P-ABORT indication primitive
SPERind	S-P-EXCEPTION-REPORT indication primitive
SPTind	S-TOKEN-PLEASE indication primitive
SRELind	S-RELEASE indication primitive
SRELcnf+	S-RELEASE (accept) confirm primitive
SRELcnf–	S-RELEASE (reject) confirm primitive
SRSYNind	S-RESYNCHRONIZE indication primitive
SRSYNcnf	S-RESYNCHRONIZE confirm primitive
SSYNMind	S-SYNC-MAJOR indication primitive
SSYNMcnf	S-SYNC-MAJOR confirm primitive
SSYNmind	S-SYNC-MINOR indication primitive
SSYNmcnf	S-SYNC-MINOR confirm primitive
STDind	S-TYPED-DATA indication primitive
SUABind	S-U-ABORT indication primitive
SUERind	S-U-EXCEPTION-REPORT indication primitive

TABLE A-2/X.215

States

Abbreviated name	Name and description
STA 01	Idle, no connection
STA 02A	Wait for S-CONNECT confirm
STA 03	Wait for S-RELEASE confirm
STA 04A	Wait for S-SYNC-MAJOR confirm
STA 04B	Wait for S-ACTIVITY-END confirm
STA 05A	Wait for S-RESYNCHRONIZE confirm
STA 05B	Wait for S-ACTIVITY-INTERRUPT confirm
STA 05C	Wait for S-ACTIVITY-DISCARD confirm
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 19	Wait for a recovery indication
STA 20	Wait for a recovery request
STA 21	Wait for S-CAPABILITY-DATA confirm
STA 22	Wait for S-CAPABILITY-DATA response
STA 713	Data transfer state

TABLE A-3/X.215

Events generated by the SS-user

Abbreviated name	Name and description
SACTDreq	S-ACTIVITY-DISCARD request primitive
SACTDrsp	S-ACTIVITY-DISCARD response primitive
SACTEreq	S-ACTIVITY-END request primitive
SACTErsp	S-ACTIVITY-END response primitive
SACTIreq	S-ACTIVITY-INTERRUPT request primitive
SACTIrsp	S-ACTIVITY-INTERRUPT response primitive
SACTRreq	S-ACTIVITY-RESUME request primitive
SACTSreq	S-ACTIVITY-START request primitive
SCDreq	S-CAPABILITY-DATA request primitive
SCDrsp	S-CAPABILITY-DATA response primitive
SCGreq	S-CONTROL-GIVE request primitive
SCONreq	S-CONNECT request primitive
SCONrsp+	S-CONNECT (accept) response primitive
SCONrsp–	S-CONNECT (reject) response primitive
SDTreq	S-DATA request primitive
SEXreq	S-EXPEDITED-DATA request primitive
SGTreq	S-TOKEN-GIVE request primitive
SPTreq	S-TOKEN-PLEASE request primitive
SRELreq	S-RELEASE request primitive
SRELrsp+	S-RELEASE (accept) response primitive
SRELrsp–	S-RELEASE (reject) response primitive
SRSYNreq(a)	S-RESYNCHRONIZE (abandon) request primitive
SRSYNreq(r)	S-RESYNCHRONIZE (restart) request primitive
SRSYNreq(s)	S-RESYNCHRONIZE (set) request primitive
SRSYNrsp	S-RESYNCHRONIZE response primitive
SSYNMreq	S-SYNC-MAJOR request primitive
SSYNMrsp	S-SYNC-MAJOR response primitive
SSYNmreq	S-SYNC-MINOR request primitive
SSYNmrsp	S-SYNC-MINOR response primitive
STDreq	S-TYPED-DATA request primitive
SUABreq	S-U-ABORT request primitive
SUERreq	S-U-EXCEPTION-REPORT request primitive

TABLE A-4/X.215

Operations on variables

Events	Condition for valid 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
SSYNMind SACTEind		if Vsc true	unchanged	V(M) + 1	unchanged	unchanged
		if Vsc false	set to V(M)	V(M) + 1	unchanged	unchanged
SSYNmind		if Vsc true	unchanged	V(M) + 1	unchanged	true
		if Vsc false	set to V(M)	V(M) + 1	unchanged	true
SSYNMrsp SACTErsp	sn = V(M) − 1		set to V(M)	unchanged	set to V(M)	unchanged
SSYNMcnf SACTEcnf			set to V(M)	unchanged	set to V(M)	unchanged
SSYNmrsp	Vsc = true and V(M) > sn > = V(A) *		set to sn + 1	unchanged	unchanged	unchanged
SSYNmcnf	Vsc = false and V(M) > sn > = V(A)		set to sn + 1	unchanged	unchanged	unchanged
SRSYNreq	r:V(M) > = sn > = V(R)		unchanged	unchanged	unchanged	unchanged
SRSYNind		abandon restart set	unchanged unchanged unchanged	set to sn unchanged unchanged	unchanged unchanged unchanged	unchanged unchanged unchanged
SRSYNrsp SRSYNcnf	a: sn as in SRSYNind r: sn as in SRSYNind s: sn < = 999999	abandon restart set	set to sn set to sn set to sn	set to sn set to sn set to sn	0 unchanged 0	unchanged unchanged unchanged
SACTRreq SACTRind			set to sn + 1	set to sn + 1	set to 1	unchanged
SACTSreq SACTSind			set to 1	set to 1	set to 1	unchanged
SCONrsp+ SCONcnf+		sn present	set to sn	set to sn	0	false

sn: synchronization point serial number quoted in session service primitive.
> =: 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.215

Specific actions

[5]	Set V(A) = V(M) = serial number in S-CONNECT response or S-CONNECT confirm Set V(R) = 0 Set Vcoll = false Set Vrsp = no Set Vsc = false Set FU(f) for f in fu-dom according to session user requirements in S-CONNECT response or S-CONNECT confirm If FU(ACT) = true, set Vact = false, set Vdnr = false
[11]	Update the position of the tokens
[12]	Set Vact = true
[14]	Set Vact = false
[16]	Update Vrsp and, if necessary, Vrspnb
[17]	Set Vrsp = no
[18]	Set Vcoll = true
[19]	Set V(M) = serial number
[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
[30]	Set the position of the tokens such that all available tokens are not owned. Set Vact = false
[31]	If Vsc false, set V(A) = V(M) Set V(M) = V(M) + 1
[32]	Set Vdnr = true

TABLE A-6/X.215

Predicates

p03	I(dk)
p04	FU(FD) & ^Vcoll
p06	FU(TD)
p07	FU(TD) & ^Vcoll
p08	FU(EX)
p09	FU(EX) & ^Vcoll
p10	^Vcoll
p11	II(ma)
p13	(^FU(ACT) OR Vact) & I(dk) & I(mi) & II(ma)
p15	(^FU(ACT) OR Vact) & I(dk) & II (mi)
p18	(^FU(ACT) OR Vact) & FU(SY) & Vsc
p20	serial number = V(M) − 1
p21	V(M) > serial number > = V(A)
p25	(FU(SY) OR FU(MA)) & FU(RESYN)
p26	(^FU(ACT) OR Vact)
p28	FU(RESYN)
p29	(^FU(ACT) OR Vact) & FU(RESYN)
p32	serial number > = V(R)
p33	V(M) > = serial number > = V(R)
p34	FU(ACT)
p39	Vact & II(ma)
p43	((Vrsp = r) & (serial number = Vrspnb)) OR ((Vrsp = a) & serial number as in SRSYNind) OR (Vrsp = s)
p45	(FU(ACT) & ^Vact) & I(dk) & I(mi) & I(ma)
p47	FU(CD) & (FU(ACT) & ^Vact) & I(dk) & I(mi) & OWNED(ma)
p50	FU(EXCEP) & (^FU(ACT) OR Vact) & AA(dk)
p51	FU(EXCEP) & (^FU(ACT) OR Vact) & II(dk)
p53	ALL(AV, RT) & RT not empty
p54	ALL(II, GT)
p55	(FU(ACT) & ^Vact) & ANY(II, tk-dom)
p57	ALL(II, GT) & (dk not in GT)
p58	ALL(II, GT) & (dk in GT)
p60	ALL(AA, GT) & (dk not in GT)
p61	ALL(AA, GT) & (dk in GT)
p63	ALL(I, tk-dom) & (^FU(ACT) OR ^Vact)
p67	FU(NR)
p69	Vcoll
p71	FU(ACT) & Vact & I(dk) & I(mi) & II(ma)
p75	(Vcoll & Vdnr) OR ^Vcoll
p81	(Vrsp = r) & ((Vrspnb > serial number) OR ((Vrspnb = serial number) & p95))
p82	(Vrsp = r) OR (p95 & p99)
p83	(Vrsp = s) OR p82
p95	SS-user is initiator of the session connection
p99	Resynchronize Type parameter in SRSYNreq is equal to Vrsp

TABLE A-7/X.215

Connection establishment state table

STATE / EVENT	STA01 idle disconnected	STA02A await SCONcnf	STA08 await SCONrsp
SCONcnf+		[5] [11] STA713	
SCONcnf−		STA01	
SCONind	STA08		
SCONreq	STA02A		
SCONrsp+			[5] [11] STA713
SCONrsp−			STA01

TABLE A-8/X.215

Data transfer state table

STATE / EVENT	STA03 await SRELcnf	STA04A await SSYNMcnf	STA04B await SACTEcnf	STA09 await SRELrsp	STA10A await SSYNMrsp	STA10B await SACTErsp	STA713 data transfer
SDTind	STA03	STA04A	STA04B				STA713
SDTreq				p04 STA09	p03 STA10A	p03 STA10B	p03 STA713
SEXind	STA03	STA04A	STA04B				STA713
SEXreq				p09 STA09	p08 STA10A	p08 STA10B	p08 STA713
STDind	STA03	STA04A	STA04B				STA713
STDreq				p07 STA09	p06 STA10A	p06 STA10B	p06 STA713

TABLE A-9/X.215

Synchronization state table

STATE / EVENT	STA03 await SRELcnf	STA04A await SSYNMcnf	STA04B await SACTEcnf	STA09 await SRELrsp	STA10A await SSYNMrsp	STA10B await SACTErsp	STA713 data transfer
SACTEcnf			[14] [22] STA713				
SACTEind							[31] STA10B
SACTEreq							p71 [24] STA04B
SACTErsp						[14] [22] STA713	
SSYNMcnf		[22] STA713					
SSYNMind							[31] STA10A
SSYNMreq							p13 [24] STA04A
SSYNMrsp					[22] STA713		
SSYNmcnf	[25] STA03	[25] STA04A	[25] STA04B				[25] STA713
SSYNmind							[23] STA713
SSYNmreq							p15 [24] STA713
SSYNmrsp				p18&p21 [25] STA09	p18&^p20&p21 [25] STA10A	p18&^p20&p21 [25] STA10B	p18&p21 [25] STA713

TABLE A-10/X.215

Resynchronization state table

EVENT \ STATE	STA03 await SRELcnf	STA04A await SSYNMcnf	STA04B await SACTEcnf	STA05A await SRSYNcnf	STA09 await SRELrsp	STA10A await SSYNMrsp
SRSYNcnf				[28] STA713		
SRSYNind(a)	[16] [19] STA11A	[16] [19] STA11A	[16] [19] STA11A	[16] [19] STA11A		[16] [19] STA11A
SRSYNind(r)	[16] STA11A	[16] STA11A	[16] STA11A	[16] STA11A		
SRSYNind(s)	[16] STA11A	[16] STA11A	[16] STA11A	[16] STA11A		[16] STA11A
SRSYNreq(a)		p28 [16] STA05A			p10& p28&^p34 [16] STA05A	p28 [16] STA05A
SRSYNreq(r)					p10& p25&^p34&p33 [16] STA05A	p25&p33 [16] STA05A
SRSYNreq(s)		p28 [16] STA05A			p10& p25&^p34 [16] STA05A	p25 [16] STA05A
SRSYNrsp						

TABLE A-10/X.215 (*concluded*)

Resynchronization state table

EVENT \ STATE	STA10B await SACTErsp	STA11A await SRSYNrsp	STA19 await recovery indication	STA20 await recovery request	STA713 data transfer
SRSYNcnf					
SRSYNind(a)			[16] [19] STA11A	[16] [19] STA11A	[16] [19] STA11A
SRSYNind(r)			[16] STA11A	[16] STA11A	[16] STA11A
SRSYNind(s)			[16] STA11A	[16] STA11A	[16] STA11A
SRSYNreq(a)	p28 [16] STA05A	p83 [16] STA05A		p28 [16] STA05A	p29 [16] STA05A
SRSYNreq(r)	p25&p33 [16] STA05A	p81&p33 [16] STA05A		p25&p33 [16] STA05A	p25&p26&p33 [16] STA05A
SRSYNreq(s)	p25 [16] STA05A	p82 [16] STA05A		p25 [16] STA05A	p25&p26 [16] STA05A
SRSYNrsp		p43 [28] STA713			

TABLE A-11/X.215

Activity interrupt and discard state table

STATE / EVENT	STA04A await SSYNMcnf	STA04B await SACTEcnf	STA05A await SRSYNcnf	STA05B await SACTIcnf	STA05C await SACTDcnf	STA10A await SSYNMrsp	STA10B await SACTErsp
SACTDcnf					[29] STA713		
SACTDind			STA11C			STA11C	STA11C
SACTDreq	p34&p39 STA05C	p39 STA05C					
SACTDrsp							
SACTIcnf				[29] STA713			
SACTIind			STA11B			STA11B	STA11B
SACTIreq	p34&p39 STA05B	p39 STA05B					
SACTIrsp							

TABLE A-11/X.215 (*concluded*)

Activity interrupt and discard state table

STATE / EVENT	STA11A await SRSYNrsp	STA11B await SACTIrsp	STA11C await SACTDrsp	STA19 await recovery indication	STA20 await recovery request	STA713 data transfer
SACTDcnf						
SACTDind				STA11C	STA11C	STA11C
SACTDreq	p34&p39 STA05C				p34&p11 STA05C	p34&p39 STA05C
SACTDrsp			[30] STA713			
SACTIcnf						
SACTIind				STA11B	STA11B	STA11B
SACTIreq	p34&p39 STA05B				p34&p11 STA05B	p34&p39 STA05B
SACTIrsp		[30] STA713				

TABLE A-12/X.215

Activity start, resume and capability data state table

STATE / EVENT	STA21 await SCDcnf	STA22 await SCDrsp	STA713 data transfer
SACTRind			[12] [27] STA713
SACTRreq			p45 [12] [27] STA713
SACTSind			[12] [26] STA713
SACTSreq			p45 [12] [26] STA713
SCDcnf	STA713		
SCDind			STA22
SCDreq			p47 STA21
SCDrsp		STA713	

TABLE A-13/X.215

Token management and exceptions state table

EVENT \ STATE	STA03 await SRELcnf	STA04A await SSYNMcnf	STA04B await SACTEcnf	STA09 await SRELrsp	STA10A await SSYNMrsp	STA10B await SACTErsp
SCGind						
SCGreq						
SGTind		[11] STA04A	[11] STA04B		[11] STA10A	[11] STA10B
SGTreq		p54 [11] STA04A	p54 [11] STA04B		p54 [11] STA10A	p54 [11] STA10B
SPERind	STA20	p03 STA20 ^p03 STA713	p03 STA20 ^p03 STA713			
SPTind	STA03	STA04A	STA04B			
SPTreq				p53 STA09	p53 STA10A	p53 STA10B
SUERind	STA20	p03 STA20 ^p03 STA713	p03 STA20 ^p03 STA713			
SUERreq				p50 STA19	p50 STA19	p50 STA19

TABLE A-13/X.215 (*concluded*)

Token management and exceptions state table

STATE / EVENT	STA19 await recovery indication	STA20 await recovery request	STA21 await SCDcnf	STA22 await SCDrsp	STA713 data transfer
SCGind					[11] STA713
SCGreq					p55 [11] STA713
SGTind	p60 [11] STA19 p61 [11] STA713	p60 [11] STA20 p61 [11] STA713	[11] STA21		[11] STA713
SGTreq		p57 [11] STA20 p58 [11] STA713			p54 [11] STA713
SPERind	STA19		STA20		p50 STA713 p51 STA20
SPTind			STA21		STA713
SPTreq				p53 STA22	p53 STA713
SUERind	STA19				p50 STA713 p51 STA20
SUERreq					p50 STA19

TABLE A-14/X.215

Connection release state table

STATE / EVENT	STA03 await SRELcnf	STA09 await SRELrsp	STA713 data transfer	Any other state
SPABind	STA01	STA01	STA01	STA01
SRELcnf+	STA01	[32] STA09		
SRELcnf−	STA713			
SRELind	[18] STA09		STA09	
SRELreq			p63 STA03	
SRELrsp+		p75 STA01 p69&p95 STA03		
SRELrsp−		p67 STA713		
SUABind	STA01	STA01	STA01	STA01
SUABreq	STA01	STA01	STA01	STA01

ANNEX B

(to Recommendation X.215)

Usage of the generalized OSI session service to achieve compatibility with Basic T.62

B.1 *Compatibility requirements*

Recommendation T.62 for the Basic Teletex Service was adopted by the CCITT in 1980 and has been used in a number of products available or under development.

It is of essential importance that the existing and future Telematic terminals and systems must be able to interact with OSI-Systems.

This was one of the main guidelines for the development of the generalized OSI session protocol which has been conducted in very close cooperation between CCITT and ISO during the last two years of the 1981-84 study period.

This Annex shows how compatibility between OSI generalized session protocol and the basic T.62 can be achieved.

B.2 *Principles for ensuring compatibility*

The compatibility requirements outlined in the previous section impose that systems using OSI protocols can interact with terminals using Telematic protocols.

The layered structure of both protocols which conforms with the OSI Reference Model and the full compatibility of the transport-oriented layer protocols limits the problem of compatibility to the protocols above the transport-oriented layers.

As far as the higher layer protocols are concerned, the principles used to ensure the required compatibility are the following:

a) The functions of T.62 usable directly in a generalized session protocol are identified.

b) These functions and the corresponding protocol elements are integrated in the generalized OSI session protocol which must remain consistent and to continue to satisfy the various requirements of a generalized OSI session protocol.

c) The additional matters in T.62 related to the particular services and to T.61 are placed in the presentation and application layer on the top of the OSI session protocol. With appropriate application rules using the session services, the T.62 protocol elements can be generated.

The usage rules of the generalized session service are described in the next section followed by the explanation of how these translate precisely in basic T.62, thus ensuring the required compatibility.

B.3 *Usage session service to ensure compatibility with basic T.62*

The following rules specify how an OSI session user entity has to use the generalized session service to give full T.62 basic Teletex/Facsimile service (see also Figures B-1/X.215 and B-2/X.215).

The BAS subset of the generalized session service must be used.

Only the following service primitives must be used:

S-CONNECT
S-RELEASE
S-U-ABORT
S-P-ABORT
S-ACTIVITY-START
S-ACTIVITY-RESUME
S-ACTIVITY-END
S-ACTIVITY-INTERRUPT
S-ACTIVITY-DISCARD
S-SYNC-MINOR
S-U-EXCEPTION-REPORT
S-P-EXCEPTION-REPORT
S-CONTROL-GIVE
S-TOKEN-PLEASE
S-CAPABILITY DATA
S-DATA

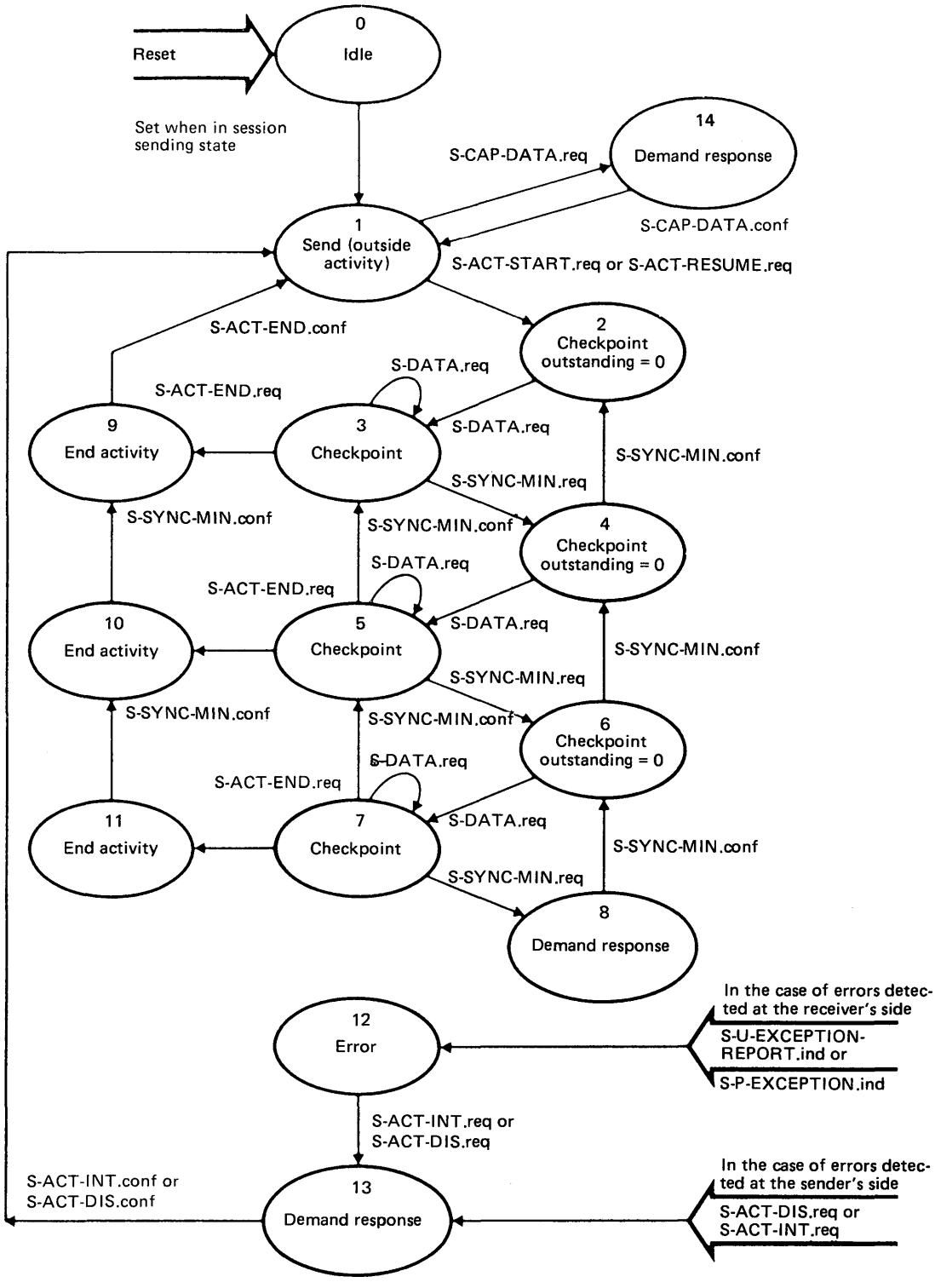
The data taken must be available.

Note – Minor sync and major/activity tokens are always available in BAS. The release token is not available.

B.3.1 *Session connection establishment phase*

The session service user initiates a session connection by using the S-CONNECT request primitive.

The acceptor SS-user may accept or refuse the connection with the S-CONNECT response primitive. When accepting the connection the SS-USER may request the control by issuing a S-TOKEN-PLEASE request primitive (see § B.3.5).

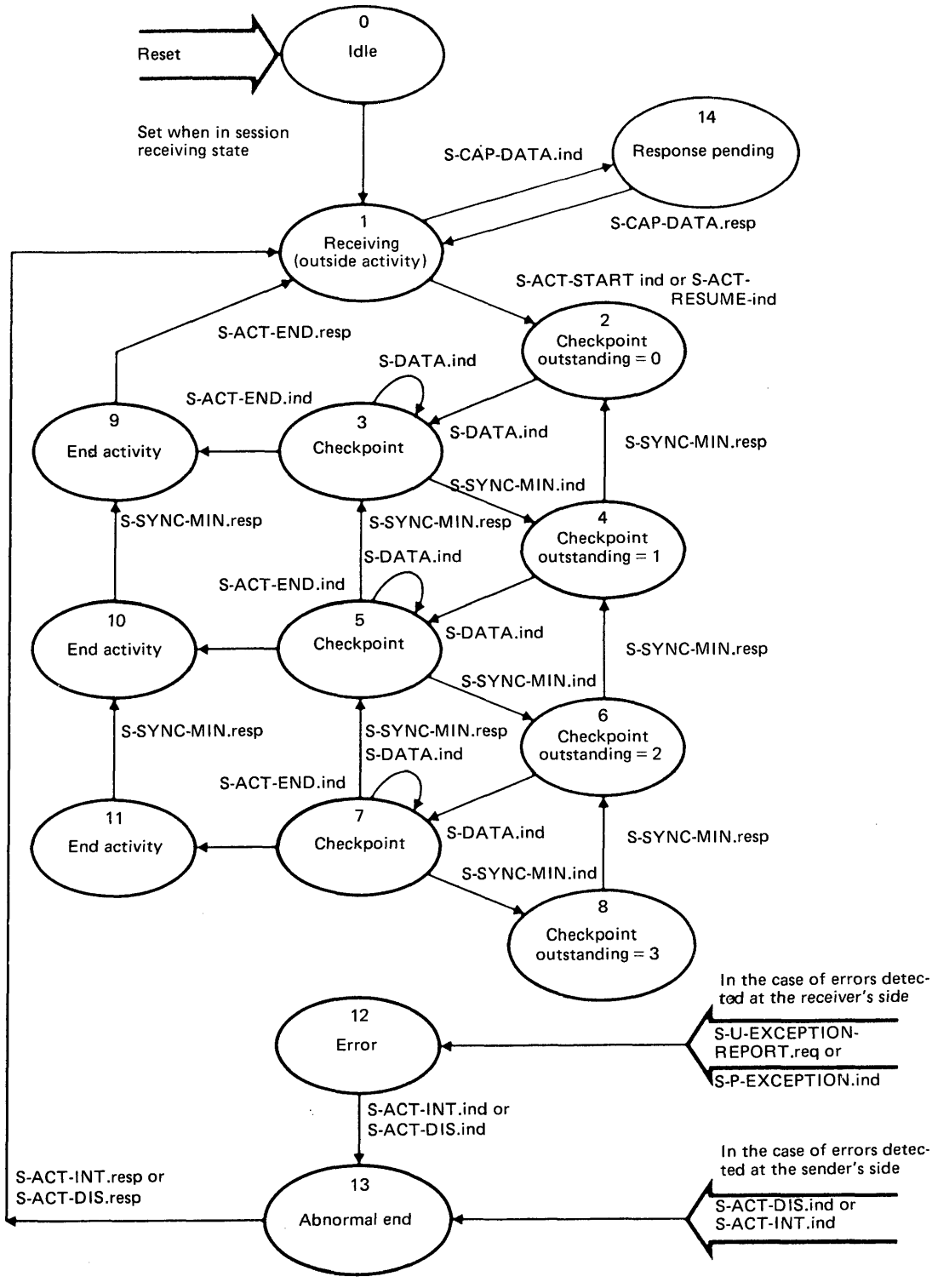


Note – This diagram assumes a window size of 3.

CCITT - 36390

FIGURE B-1/X.215

Diagram showing Session Service Interactions for the sending Session Service User



Note – This diagram assumes a window size of 3.

FIGURE B-2/X.215

Diagram showing Session Service Interactions for the receiving Session Service User

Parameters of the S-CONNECT primitives are used in the following way:

- Connection identifier: supplied by the user in the request and response, its format is the same as defined in T.62 (calling or called terminal identifier, date and time, additional session reference number).

  Note – The four fields of the connection identifier have lengths as identified in Recommendation F.200.

- Calling and called SSAP addresses: the session layer addressing is not used by T.62 equipments.
- Quality of service:
  - no extended control (i.e. no transport expedited)
  - no optimized dialogue transfer (i.e. basic concatenation)
  - other parameters, if available, must be set in order to select class 0 of the transport protocol.
- Session requirements: the following functional units have to be selected:
  - Synchronization minor
  - Activity management
  - Capability data
  - Half duplex
  - Exceptions.
- Initial assignment of tokens: all the available tokens are assigned to the initiator.
- User data: this parameter contains the following information (see also § B.5):
  - miscellaneous session capabilities
  - window size (to be negotiated according to the rules defined in T.62)
  - service identifier
  - non basic terminal capabilities.
- Result: (in response/confirmation) this parameter is used to accept or refuse the session connection.

  In case of refusing the connection, a Reason (session) information may be present in the user data.

B.3.2 *Session connection termination phase*

The session connection can be terminated by means of the S-RELEASE primitives (orderly release) or S-U-ABORT/S-P-ABORT primitives.

Only the initiator of the session connection is allowed to release it by using S-RELEASE request.

No user data have to be supplied and the result parameter in response/confirmation indicates acceptation of the release (since the release token is not available, there is no way to refuse the release).

When requesting S-U-ABORT, the user data parameter must be absent.

Re-use of the transport connection is a local implementation choice, and does not appear at the session service level.

B.3.3 *Document management*

The T.62 document concept is equivalent to the BAS activity concept where the activity identifier is the document number.

The SS-user will use the S-ACTIVITY-START and S-ACTIVITY-RESUME primitives to start or resume documents; when resuming a previously interrupted document it is up to the user to supply and control the linking informations.

The SS-user terminates a document by using the S-ACTIVITY-END confirmed service.

Documents may be interrupted or discarded by using the S-ACTIVITY-INTERRUPT or S-ACTIVITY-DISCARD services.

B.3.3.1 *S-ACTIVITY-START*

- Activity identifier: this parameter contains the document reference number.
- User data: this parameter contains the non basic terminal capabilities, the document type identifier, the service interworking identifier encoded as defined in T.62.

B.3.3.2 *S-ACTIVITY-RESUME*

- Activity identifier: this parameter contains the document reference number.
- Old activity identifier is the same activity identifier that had been supplied when the activity had been started.
- The old session connection identifier identifies the session in which the activity had been started, it must contain the calling terminal identifier, the called terminal identifier, date and time and additional session reference number.
- The user data parameter must contain the non basic terminal capabilities, the document type identifier, the service interworking identifier encoded as in T.62.

Note – It is the responsibility of the receiving terminal to discard any user information that has been duplicated in the process of continuation of an interrupted activity.

B.3.3.3 *S-ACTIVITY-END*

- Synchronization point serial number (request/indication): the last checkpoint reference number of the document.
- No user data are supplied.

The SS-user must not use S-ACTIVITY-END request immediately after having requested a minor synchronization point (in T.62 data must be sent between the last page boundary and the end of the document).

To refuse the checkpoint indicated in S-ACTIVITY-END indication, the SS-user shall use the U-USER-EXCEPTION-REPORT service.

When activating the S-ACTIVITY-END response primitive, the receiving SS-user indicates that:

- it has not detected any error
- it accepts responsibility for the received document
- it is ready to receive a new S-ACTIVITY-START or S-ACTIVITY-RESUME indication.

B.3.3.4 *S-ACTIVITY-DISCARD*

The S-ACTIVITY-DISCARD request primitive shall be used to indicate to the remote entity, the abnormal ending of a document and that the receiver of S-ACTIVITY-DISCARD indication is not held responsible for the part of the document received so far.

Note – S-ACTIVITY-DISCARD indication is an invitation to discard the whole of the document.

The S-ACTIVITY-DISCARD response primitive shall be used to acknowledge the S-ACTIVITY-DISCARD indication and to indicate that the SS-user is ready to receive a new S-ACTIVITY-BEGIN indication.

The SS-user may use the reason parameter in the S-ACTIVITY-DISCARD primitives, only the following reasons shall be indicated:

a) local terminal error

b) unrecoverable procedural error

c) no specific reason stated.

B.3.3.5 *S-ACTIVITY-INTERRUPT*

The S-ACTIVITY-INTERRUPT request shall be used to indicate to the remote entity the point of resynchronization and to abnormally end the document transfer in progress.

The S-ACTIVITY-INTERRUPT response primitive shall be used to acknowledge the S-ACTIVITY-INTERRUPT indication. It confirms to the initiator of S-ACTIVITY-INTERRUPT that the receiver has already accepted responsibility for the received document (up to the last synchronization point for which a positive acknowledgement has been sent).

Linking of parts of an interrupted document is a local operation at the receiver and is therefore not within the responsibility of the session service provider. The SS-user may use the reason parameter in the S-ACTIVITY-INTERRUPT primitive, only one of the following reasons shall be indicated:

a) local terminal error

b) unrecoverable procedural error

c) no specific reason stated.

B.3.4 *Synchronization*

The SS-user will not request major synchronization (since the S-SYNC-MAJOR primitive has not been selected during session connection establishment phase).

In the basic Teletex service, a minor synchronization point must be inserted at each page boundary using the S-SYNC-MINOR request primitive.

The user will use only the minor synchronization service with explicit confirmation requested.

The SS-user must not request an end of activity or a minor synchronization point immediately after having requested a minor synchronization point.

The maximum window size may be negotiated during session connection establishment by using the user data parameter of the S-CONNECT primitive. (The negotiation rules are defined in T.62).

The sender (i.e. the owner of all the tokens) is permitted to recover from an interrupted transmission at only one of two points:

a) A cancellation is achieved by the subsequent use of S-ACTIVITY-RESUME request and S-ACT-DISCARD request and the transmission will be resumed by the S-ACTIVITY-START request.

b) The sender may resume by use of S-ACTIVITY-RESUME starting at the last minor sync point for which an S-SYNC-MINOR confirmation was received.

On this basis, the receiver must be able to resume reception at a minor sync point ranging from the last acknowledged sync point, to the last acknowledged sync point plus one minus the window size.

The S-SYNC-MINOR request primitive is used to indicate the boundary between pages and it also indicates a checkpoint for error recovery purposes. The S-SYNC-MINOR indication invites the SS-user to accept responsibility for the previously received page.

The S-SYNC-MINOR response shall be used to indicate that the user accepts responsibility for the page and acknowledges the minor sync point. Each minor sync point must be explicitly acknowledged.

Note – The rules (i.e. the state machine) to use the session service for synchronization are unaffected by the inclusion or exclusion of the synchronization window mechanism within/from the session layer.

B.3.4.1 *S-SYNC-MINOR*

– Type (request/indication): this parameter must indicate that explicit confirmation is requested.

– Synchronization point serial number: (indication/response/confirm) check point number.

– No user data must be supplied in the request.

– The user data must be supplied in the response with the first octet encoded as follows:

0 further traffic can be accepted
1 ability to receive further traffic is jeopardized.

B.3.4.2 *S-U-EXCEPTION-REPORT*

– Reason:

a) SS-user receiving ability jeoparidized
b) local SS-user error
c) sequence error
d) unrecoverable procedural error
e) non-specific error

– User data parameter must not be supplied.

The receiver of a document may issue an S-U-EXCEPTION-REPORT request at any time after having received an S-SYNC-MINOR indication, or S-ACTIVITY-END indication instead of giving the confirmation.

When receiving an S-U-EXCEPTION-REPORT indication or S-P-EXCEPTION-REPORT indication, the user must request the S-ACTIVITY-INTERRUPT or S-ACTIVITY-DISCARD services.

B.3.5 *Control exchange*

The S-CONTROL-GIVE service is used to give all the available tokens. This can only be used when no activity is in progress. The control give service is unconfirmed (although it is confirmed at the protocol level).

B.3.5.1 *S-TOKEN-PLEASE*

Must only be used to request all the available tokens by requesting only the data token.

When receiving an S-TOKEN-PLEASE indication with data token parameter, the SS-USER may give the control by requesting the give control service.

B.3.5.2 *S-CONTROL-GIVE*

There is no parameter associated with these service primitives.

B.3.5.3 *S-TOKEN-PLEASE*

Tokens: data token only to demand control.

No user data.

B.3.6 *Data transfer phase*

The S-DATA service must be used to send user information only within an activity.

B.3.7 *Capability exchange*

The document capability list is exchanged by using the S-CAPABILITY-DATA primitives.

The list of the capability parameters is described in T.62, § 3.4.5, and these parameters are supplied by means of the user data parameter of the S-CAPABILITY-DATA primitives.

B.4 *Usage of the session service to ensure compatibility with non basic T.62*

The following rules specify how an OSI session service user has to use the generalized session service to ensure compatibility with non basic T.62 (i.e. extension for interactive mode, typed data facility and duplex exchanges).

The rules for ensuring compatibility with basic T.62 remain unchanged except for the services described in this section.

The following additional service primitives may be used:

S-TOKEN-GIVE
S-TYPED-DATA

B.4.1 *Connection establishment phase*

For pure interactive mode, the BCS subset must be selected.

For interactive mode with document transfer, the BAS subset must be selected.

The SS-user indicates in the session requirements parameter which of the duplex and half duplex functional units is selected.

He may optionally propose the use of the typed data functional unit.

B.4.2 *Tokens exchange*

Tokens are never exchanged when an activity is in progress.

The S-CONTROL-GIVE service can be used to give all the available tokens outside activities. This service may not be used if pure interactive mode has been selected at the session establishment phase.

The S-GIVE-TOKEN service must be used to provide interactive token exchange (i.e. unconfirmed at the protocol level). All the available tokens must be given when invoking this service. The S-PLEASE-TOKEN service is used to request all the available tokens (by requesting either data, sync minor and major tokens or only the data token).

B.4.3 *Data transfer*

When interactive mode with document transfer is selected, data may be sent inside and outside activities.

When the duplex functional unit is selected, data may be sent by both users outside activities. Only the SS-user who owns the minor sync token and major/activity token is allowed to send data when an activity is in progress.

S-TYPED-DATA service may be used if the corresponding functional unit has been selected at connection establishment.

B.4.4 *Exception reporting*

In the case of an error occurring during an interactive phase (i.e. outside an Activity in BAS, or in BCS), only the S-U-ABORT service can be used to resolve it.

B.5 *Handling of T.62 PIs and PGIs not defined within the OSI session layer*

Recommendation T.62 defines a number of PIs and PGIs which are not part of the OSI session layer. Some of them are considered as components of valid session SPDUs. For example, Calling Terminal Identifier, Date and Time and Additional Session Reference Number are components of the Session Connection Identifier Parameter of the CONNECT SPDU.

Although the others are recognized within the session layer specification, they are not defined in the OSI session protocol. Therefore, local conventions are needed in order to ensure that the corresponding protocol elements are generated and received in accordance with Recommendation T.62.

The SPDUs and parameters which are subject to such conventions are listed in Table B-1/X.215.

TABLE B-1/X.215

S.62 Parameters subject to local conventions

	CONNECT	ACCEPT	REFUSE	ACTIVITY-START	ACTIVITY-RESUME	CAPABILITY DATA	CAPABILITY DATA ACK
Non-basic session capabilities	X	X	X				
Service identifier	X	X	X				
Inactivity timer	X	X				X	X
Service interworking identifier				X	X		
Acceptance of CDCL parameters							X
Storage capability negotiation						X	X
Document type identifier				X	X		
Non-basic teletex terminal capabilities	X	X	X	X	X	X	X

Recomendación X.216

DEFINICIÓN DEL SERVICIO DE PRESENTACIÓN PARA LA INTERCONEXIÓN DE SISTEMAS ABIERTOS PARA APLICACIONES DEL CCITT[1)]

(Melbourne, 1988)

El CCITT,

considerando

(a) que la Recomendación X.200 define el modelo de referencia de interconexión de sistemas abiertos para aplicaciones del CCITT;

(b) que la Recomendación X.210 define los convenios relativos a las definiciones de los servicios de las capas del modelo de referencia de ISA;

(c) que la Recomendación X.215 define el servicio de sesión para la interconexión de sistemas abiertos para aplicaciones del CCITT;

---

1) La Recomendación X.216 y la norma ISO 8822 [Information processing systems – Open systems interconnection – Connection oriented presentation service definition] fueron desarrolladas en estrecha colaboración y están técnicamente armonizadas.

(d) that Recommendation X.220 specifies the use of X.200 series protocols in CCITT applications;

(e) that Recommendation X.410 – 1984 specifies the protocol for Remote Operation and Reliable Transfer Server for Message Handling Systems;

(f) that Recommendation X.226 specifies the Presentation Protocol Specification for Open Systems Interconnection for CCITT applications,

unanimously declares

that this Recommendation defines the Presentation Service of Open Systems Interconnection for CCITT Applications as given in the Scope and Field of Application.

CONTENTS

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.

The aim of Open Systems Interconnection is to allow, with a minimum of technical agreement outside the interconnection recommendations, the interconnection of information processing systems

- from different manufacturers;
- under different managements;
- of different levels of complexity;
- of different ages.

This Recommendation defines the service available to entities within the Application Layer of the Reference Model.

This Recommendation recognizes that application-entities may wish to intercommunicate for a wide variety of reasons. While not all systems will share a common method of representing the information they wish to intercommunicate, they will be agreed about the subject matter of their communication and the meanings to be assigned to that information. The presentation-service provides the proper means of transferring information so that the semantics are preserved during the transfer.

It is recognized that, with respect to presentation quality of service (QOS) described in § 9, 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 tha the individual treatments in each layer satisfy overall QOS objectives in a consistent manner. As a consequence, an annex may be added to this Recommendation at a later time which reflects further QOS developments and integration.

1 Scope and field of application

1.1 This Recommendation defines (in an abstract way) the externally visible service provided by the OSI Presentation Layer in terms of

a) the primitive actions and events at the user/service boundary;

b) the parameter data associated with each primitive action and event;

c) the relationship between, and the valid sequences of, those actions and events.

1.2 The service defined in this Recommendation is that which is provided by an OSI presentation protocol (in conjunction with the OSI session-service) and which may be used by any OSI application protocol.

1.3 This Recommendation does not specify individual implementations or products, nor does it constrain the implementation of entities and interfaces within a computer system. There is, therefore, no conformance to this Recommendation.

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 8509).
ISO 7498-3	Information processing systems – Open Systems Interconnection – Basic Reference Model – Part 3 Naming and addressing[2].
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.226	Presentation Protocol Specification for Open Systems Interconnection for CCITT application (see also ISO 8823).
Recommendation X.410-1984	Message Handling Systems: Remote Operation and Reliable Transfer Server.

[2] At present at the stage of draft.

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-entity;
b) application-protocol-control-information;
c) presentation-connection;
d) presentation-entity;
e) Presentation Layer;
f) presentation-service;
g) presentation-service-access-point;
h) presentation-service-data-unit;
i) session-connection;
j) session-service;
k) transfer syntax;
l) concrete syntax;
m) real open system.

Note – The abbreviations in § 4 apply to some of these terms.

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;
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) calling-presentation-address;
b) called-presentation-address;
c) responding-presentation-address.

3.4 *Presentation-service definitions*

For the purpose of this Recommendation the following definitions apply:

3.4.1 **destructive**: A service is destructive if its invocation may cause loss of undelivered data of other service primitives.

3.4.2 **non-destructive**: A service is non-destructive if its invocation does not cause the loss of data.

3.4.3 **abstract syntax**: The specification of Application Layer data or application-protocol-control-information by using notation rules which are independent of the encoding technique used to represent them.

3.4.4 **abstract syntax name**: A name which unambiguously identifies an abstract syntax.

3.4.5 **transfer syntax name**: A name which unambiguously identifies either a transfer syntax or a set of rules for generating a transfer syntax from a given abstract syntax.

3.4.6 **presentation data value**: The unit of information specified in an abstract syntax, which is transferred by the presentation-service.

3.4.7 **presentation context**: An association of an abstract syntax with a transfer syntax.

Note 1 – From the viewpoint of the presentation-service-user, a presentation context represents an environment in which the presentation data values of the abstract syntax can be transferred (as a bitstring) without ambiguity.

Note 2 – Where the abstract syntax permits it, a presentation data value may contain embedded fields, each of which carries a presentation data value from a (possibly different) abstract syntax.

Note 3 – From the viewpoint of the presentation-service-user, a presentation context represents a specific use of an abstract syntax. Multiple presentation contexts may be defined for the same abstract syntax (with the same or different transfer syntaxes); presentation data values transmitted in these separate presentation contexts are also delivered in these separate presentation contexts.

3.4.8 **defined context set**: A set of presentation contexts that has been defined by agreement between all three parties to a communication: i.e. the presentation-service-provider and two presentation-service-users.

Note – The inclusion of a presentation context in the defined context set implies that its abstract syntax is acceptable to both presentation-service-users and that the cooperating presentation-entities have agreed on an acceptable transfer syntax for that presentation context.

3.4.9 **inter-activity defined context set**: A set of presentation contexts which is defined for a presentation-connection when the (session) activity management functional unit is selected. It initially takes the value of the defined context set at presentation-connection establishment, and is further modified only by P-ALTER-CONTEXT service primitives issued outside of activities.

3.4.10 **default context**: The default context is a presentation context which is always known to the presentation-service-provider and two presentation-service-users for a given presentation-connection. It is the presentation context which always applies to the User data parameter of the P-EXPEDITED-DATA service primitives. It applies to the User data parameters of other service primitives only when the defined context set is empty.

Note – The use of an implied default context can arise when no name for default context is specified.

3.4.11 **functional unit**: A logical grouping of services defined by this Recommendation for the purpose of

- negotiation during the presentation-connection establishment, for use on the presentation-connection;
- referencing by other standards.

3.4.12 **disrupt**: A service procedure is disrupted by another service if the second service results in service primitives of the first service not being used as specified for the procedure of the first service.

3.4.13 **X.410-1984 mode**: A restricted mode of operation of the Presentation Layer, which is used to allow interworking with a system that conforms to CCITT Recommendation X.410 (1984).

3.4.14 **normal mode**: The mode of operation of the Presentation Layer, which provides the full facilities of the presentation-service.

3.4.15 **initiator**: The presentation-entity or presentation-service-user that initiates the presentation-connection establishment.

3.4.16 **responder**: The presentation-entity or presentation-service-user that responds to a presentation-connection establishment proposal.

3.4.17 **requestor**: The presentation-entity or presentation-service-user that initiates a particular action.

3.4.18 **acceptor**: The presentation-entity or presentation-service-user that accepts a particular action.

3.4.19 **presentation context identification**: The identification of a specific presentation context at the conceptual service boundary.

4 Abbreviations

ASN.1 Abstract Syntax Notation One (see Recommendation X.208)
DCS Defined Context Set
PCEP presentation-connection-end-point
PS presentation-service
PSAP presentation-service-access-point
PS-user presentation-service-user
SS session-service

5 Conventions

This Recommendation uses the descriptive conventions defined in Recommendation X.210.

6 Overview of the Presentation Service

6.1 *Purpose*

The Presentation Layer is concerned with the presentation of information in transit between open systems (see Recommendation X.200).

6.2 *Relationship to Application Layer*

Note – The Presentation Layer view of the Application Layer is described below.

6.2.1 An application protocol is specified in terms of the transfer of presentation data values between application-entities (PS-users), using the User data parameter of presentation-service primitives.

6.2.2 A set of presentation data value definitions associated with an application protocol constitutes an abstract syntax. For two application-entities to communicate successfully they must have an agreement on the set of abstract syntaxes they intend to use. During the course of communication they may decide to modify this agreement. As a consequence, the set of abstract syntaxes in use may be changed.

6.2.3 The abstract syntax specification identifies the information content of the set of presentation data values. It does not identify the transfer syntax to be used while presentation data values are transferred between presentation-entities, nor is it concerned with the local representation of presentation data values.

6.2.4 The Presentation Layer exists to ensure that the information content of presentation data values is preserved during transfer. It is the responsibility of cooperating application-entities to determine the set of abstract syntaxes they employ in their communication and inform the presentation-entities of this agreement. Knowing the set of abstract syntaxes to be used by the application-entities, the presentation-entities are responsible for selecting mutually acceptable transfer syntaxes that preserve the information content of presentation data values.

Note – Presentation-entities have no role in determining the set of abstract syntaxes to be used by application-entities.

6.3 *Relationship to Session Layer*

Presentation-entities support protocols that enhance the OSI session-service in order to provide a presentation-service with the facilities described in Recommendation X.200. The PS-user is provided with access to the session-service which permits full use to be made of that service. This includes negotiation of and access to the session functional units. The role of the Presentation Layer in providing this access includes representation of presentation data values in the User data parameters of session-service primitives.

Note – It is not the function of the Presentation Layer to provide dialogue control and data transfer functions additional to those provided by the session-service.

6.4 *Features of the Presentation Layer*

The Presentation Layer has two functions it carries out on behalf of PS-users:

a) negotiation of transfer syntaxes;

b) transformation to and from transfer syntax.

The function of transfer syntax negotiation is supported by presentation protocols; it provides presentation context definition facilities. Transformation of syntax is a function contained within a presentation-entity and has no impact on presentation protocol design.

Note 1 – It is outside the scope of the presentation-service and presentation protocol standards to constrain or specify the abstract and transfer syntaxes supported by a particular open system. The syntaxes supported by an open system depend upon the nature of the applications in which it is involved.

Note 2 – In any real open system, presentation data values will have a local concrete syntax. Transformation to and from transfer syntax is from and to that local concrete syntax.

6.5 *Negotiation of syntax*

Negotiation of transfer syntax takes place between two presentation-entities when a PS-user provides the name of an abstract syntax for which a transfer syntax is required. The result of a successful negotiation is the association of the named abstract syntax with a compatible transfer syntax; such an association constitutes a presentation context. From the viewpoint of the PS-user, a presentation context represents a specific distinct use of an abstract syntax.

In general, there need not be a unique combination of abstract syntax and transfer syntax. It may be possible to represent a specific abstract syntax by one or more transfer syntaxes; also it may be possible to use one transfer syntax to represent more than one abstract syntax.

6.6 *Information transfer*

6.6.1 User information is carried in User data parameters of presentation-service primitives. Each User data parameter contains one or more presentation data values. The order of these presentation data values is retained in transfer.

6.6.2 A presentation data value may be structured such that it contains nested presentation data values from other presentation contexts if this is supported by the abstract syntax in use for the presentation context.

Note – The structure of User data parameters of presentation-service primitives cannot be more explicitly defined at the service level. Any interface in a real open system (if such an interface exists) will define a concrete form.

6.7 *Presentation context definition*

6.7.1 The presentation-service provides facilities for the definition of presentation contexts that match the information transfer requirements of its users. One or more presentation context definitions fully describe the information transfer requirements of users of a presentation-connection.

6.7.2 There are two services by which presentation contexts may be defined. These are the P-CONNECT and the P-ALTER-CONTEXT services. The P-ALTER-CONTEXT service also provides for the deletion of presentation contexts which are no longer required.

6.7.3 As presentation contexts are defined they are added to the DCS. The action of presentation context definition makes a presentation context available for immediate use. This enables a PS-user to identify a set of presentation contexts that are required to describe fully the flow of information between PS-users.

6.7.4 If the DCS is empty, then it is still possible to transfer presentation data values in presentation-service User data parameters; in this case all presentation data values are from the default context. Presentation data values are transferred in the default context only when the DCS is empty, or in a P-EXPEDITED-DATA service primitive. The default context may be defined using the presentation-connection establishment service (but may not be redefined by any other presentation service), or may be established by prior agreement. Presentation data values which are transferred using the P-EXPEDITED-DATA service are always from the default context.

6.8 *Management of the DCS*

If the context management functional unit is not selected, then the DCS will not change during the presentation-connection and the remainder of § 6.8 does not apply.

6.8.1 *Context management functional unit*

6.8.1.1 If the context management functional unit is selected, the DCS may change during the presentation-connection. This is accomplished by using the P-ALTER-CONTEXT service. The Presentation Layer is responsible for ensuring that the DCS is identical at both ends of a presentation-connection; therefore P-ALTER-CONTEXT is a confirmed-service. However, it is possible for certain destructive services to collide with or overtake the P-ALTER-CONTEXT service.

6.8.1.2 If a P-RESYNCHRONIZE indication service primitive is received while awaiting a P-ALTER-CONTEXT confirm service primitive, then the P-RESYNCHRONIZE service takes precedence and the P-ALTER-CONTEXT service procedure is disrupted. The DCS is indicated to the PS-user. If a P-RESYNCHRONIZE request service primitive is issued while awaiting a P-ALTER-CONTEXT confirm request service primitive, then the P-RESYNCHRONIZE service takes precedence and the P-ALTER-CONTEXT service procedure is disrupted. The DCS is indicated to the PS-user.

6.8.1.3 Interaction of the P-ACTIVITY-INTERRUPT and P-ACTIVITY-DISCARD services with the P-ALTER-CONTEXT service may cause misalignment of the DCS and subsequent transfer of data in a presentation context unknown to one of the PS-users. PS-users can avoid this situation by use of the activity token and appropriate sequencing rules.

6.8.2 *Context restoration functional unit*

6.8.2.1 The PS-user can select the context restoration functional unit. If the context restoration functional unit is not selected, the DCS may only be changed via the P-ALTER-CONTEXT service and the remainder of § 6.8 does not apply. If the context restoration functional unit is selected, the presentation-service-provider will remember the DCS at specified points during the presentation-connection. If the PS-user requests a return to one of these points, the DCS will be restored to the one active at that point.

6.8.2.2 A P-RESYNCHRONIZE (restart) or (set) to a point known to the presentation-service-provider will restore the DCS to the one known at that point. If the point specified is lower than those known to the presentation-service provider, the DCS will be restored to that defined at presentation-connection establishment. If the point specified is higher than the ones known to the presentation-service-provider or if P-RESYNCHRONIZE (abandon) is requested, the DCS will be left unchanged. If an unknown point (i.e. within the rage of known points, but not known by the presentation-service-provider) is specified, the presentation-service-provider will indicate this to the PS-user and will not alter the DCS.

6.8.2.3 The DCS outside activities is the inter-activity DCS, which is defined at presentation-connection establishment and modified by any P-ALTER-CONTEXT request service primitive issued outside an activity. When an activity is started, its initial DCS is equal to the inter-activity DCS. Subsequently P-ALTER-CONTEXT request service primitives issued inside the activity alter only the DCS of that activity.

6.8.2.4 A P-ACTIVITY-END, P-ACTIVITY-INTERRUPT or P-ACTIVITY-DISCARD causes the presentation-service-provider to restore the DCS to the inter-activity DCS.

6.8.2.5 A P-ACTIVITY-RESUME will restore the DCS to that of the specified synchronization point in the specified activity (if known by the presentation-service-provider). Since this service is non-confirmed, it is possible to receive data that is in an unknown presentation context. If this happens, a P-P-ABORT indication will be issued to both PS-users.

Note – Control of activity identifiers is a concern of the PS-user.

7 Facilities of the service

The presentation-service comprises a number of facilities. Each facility is outlined below and the services which make up each facility are identified in Table 1/X.216.

7.1 *The connection establishment facility*

The connection establishment facility provides a service which allows a PS-user to establish a presentation-connection with another PS-user. The service allows the PS-users to exchange parameters through which they may establish the characteristics of the presentation-connection in particular

a) the presentation functional units selected;

b) the initial DCS;

c) the characteristics of the session-connection;

d) the definition of the default context.

TABLE 1/X.216

Summary of presentation facilities, their services and purpose

Name of service	Type of service	Purpose
Connection establishment facility		
P-CONNECT	Confirmed	Connection establishment
Connection termination facility		
P-RELEASE	Confirmed	Connection release
P-U-ABORT	Non-confirmed	User-initiated abort
P-P-ABORT	Provider-initiated	Provider-initiated abort
Context management facility		
P-ALTER-CONTEXT	Confirmed	Context addition and deletion
Information transfer facility		
P-DATA	Non-confirmed	(see Note)
P-TYPED-DATA	Non-confirmed	(see Note)
P-EXPEDITED-DATA	Non-confirmed	(see Note)
P-CAPABILITY-DATA	Confirmed	(see Note)
Dialogue control facility		
P-TOKEN-GIVE	Non-confirmed	(see Note)
P-TOKEN-PLEASE	Non-confirmed	(see Note)
P-CONTROL-GIVE	Non-confirmed	(see Note)
P-SYNC-MINOR	Optionally confirmed	(see Note)
P-SYNC-MAJOR	Confirmed	(see Note)
P-RESYNCHRONIZE	Confirmed	(see Note)
P-U-EXCEPTION-REPORT	Non-confirmed	(see Note)
P-P-EXCEPTION-REPORT	Provider-initiated	(see Note)
P-ACTIVITY-START	Non-confirmed	(see Note)
P-ACTIVITY-RESUME	Non-confirmed	(see Note)
P-ACTIVITE-END	Confirmed	(see Note)
P-ACTIVITY-INTERRUPT	Confirmed	(see Note)
P-ACTIVITY-DISCARD	Confirmed	(see Note)

Note – The purpose of the presentation service follows that of the corresponding session service as specified in Recommendation X.215.

7.2 *The connection termination facility*

The connection termination facility provides services which allow

a) the orderly release of a presentation-connection by the PS-users in a way which is non-destructive;

b) the termination of a presentation-connection in a way which may be destructive; termination may be initiated by either of the PS-users or by the presentation-service-provider.

7.3 *The context management facility*

The context management facility provides a service which allows

a) the addition of presentation contexts to the DCS by agreement among the two PS-users and the presentation-service-provider; an identification is associated with each defined presentation context, but this identification has no significance beyond this presentation-connection;

b) the deletion of presentation contexts from the DCS.

7.4 *The information transfer facility*

The information transfer facility provides services which allow PS-users to exchange information over a presentation-connection. The services allow data with token control, data without token control, typed data, capability data and expedited data if corresponding session functional units are selected.

7.5 *The dialogue control facility*

The dialogue control facility provides services which allow token management, synchronization, resynchronization, exception reporting and activity management, if corresponding session functional units are selected. These services are mapped onto the corresponding session services. This Recommendation describes them only in respect of their relationships to and effects on other presentation services. The presentation-service, in certain cases, imposes additional constraints on the use of the services which directly invoke the session services; the use of these services also affects the states of the presentation-entities. These session services are more fully described in the Session Service Definition (see Recommendation X.215).

8 Functional units

8.1 Functional units are used by this Recommendation for the purpose of identification of PS-user requirements during presentation-connection establishment.

8.2 Two categories of functional units exist

a) *Session functional units*, as defined in Recommendation X.215, comprising:

1) the kernel functional unit;
2) the half-duplex functional unit;
3) the duplex functional unit;
4) the expedited data functional unit;
5) the minor synchronize functional unit;
6) the major synchronize functional unit;
7) the resynchronize functional unit;
8) the activity management functional unit;
9) the negotiated release functional unit;
10) the capability data functional unit;
11) the exceptions functional unit;
12) the typed data functional unit.

The selection of session functional units which may be made is subject to the constraints imposed by the session-service, see Recommendation X.215.

Note – The decision of which session functional units are to be used is made during presentation-connection establishment.

b) *Presentation functional units*, corresponding to services provided by the Presentation Layer, and comprising:

1) the kernel functional unit;
2) the context management functional unit;
3) the context restoration functional unit.

8.3 The kernel functional unit is always available and supports information transfer in whatever service primitive User data parameters of those functional units which are selected. The context management functional unit and the context restoration functional unit are optional and their use is negotiable. The context restoration functional unit shall not be selected if the context management functional unit is not selected for use on the presentation-connection.

8.4 When a session functional unit is selected by the PS-users, the corresponding presentation services and functions are made available to the PS-users.

9 Quality of service

The definition of the quality of service concept and associated parameters, as well as the way they are negotiated during the presentation-connection establishment are strictly identical with the concepts, parameters and negotiation mechanisms defined in the Session Service Definition, Recommendation X.215.

Note – Future extensions of this Recommendation may establish a use of the quality of service parameters in determining the transfer syntax to be used.

10 Presentation Service Primitives

This Recommendation uses the abstract model for a layer service defined in Recommendation X.210. The model defines the interactions between the PS-user and the presentation-service-provider which take place at the two PSAPs. Information is passed between the PS-user and the presentation-service-provider by service primitives, which may convey parameters.

Table 2/X.216 lists the presentation-service primitives by which information is transferred to and from the PS-user.

The sequencing procedures for all services are specified in § 11.

Note – For all services which carry user data, excluding P-DATA and P-TYPED-DATA, it may not be possible to exchange PS-user data, dependent on the transfer syntax in use and the SS-user data length limitation supported by the underlying session-service. The way in which the PS-user is made aware of this is a local matter.

10.1 *User data parameters*

The information in the User data parameters of the P-EXPEDITED-DATA request and indication service primitives shall always be one or more presentation data values from the default context. The information in the User data parameters of all other presentation-service primitives shall be one or more presentation data values from presentation contexts determined by the rules governining the DCS. Any embedded presentation data values shall be from presentation contexts determined by these rules. These rules are:

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 procedure for the service primitive containing the User data parameter 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 confirm service primitive is awaited 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 there shall be no User data parameter in the service primitive.

TABLE 2/X.216

Presentation service primitives

SERVICE PRIMITIVE	PARAMETER
F-CONNECT request	Calling-presentation-address Called-presentation-address Presentation context definition list Default context name Quality of service Presentation requirements Mode Session requirements Initial synchronization point serial number Initial assignment of tokens Session connection identifier User data
P-CONNECT indication	Calling-presentation-address Called-presentation-address Presentation context definition list Presentation context definition result list Default context name Quality of service Presentation requirements Mode Session requirements Initial synchronization point serial number Initial assignment of tokens Session connection identifier User data
P-CONNECT response P-CONNECT confirm	Responding-presentation-address Presentation context definition result list Default context result Quality of service Presentation requirements Session requirements Initial synchronization point serial number Initial assignment of tokens Session connection identifier Result User data
P-RELEASE request P-RELEASE indication	User data
P-RELEASE response P-RELEASE confirm	Result User data
P-U-ABORT request P-U-ABORT indication	User data
P-P ABORT indication	Provider reason

TABLE 2/X.216 *(continued)*

SERVICE PRIMITIVE	PARAMETER
P-ALTER-CONTEXT request	Presentation context addition list Presentation context deletion list User data
P-ALTER-CONTEXT indication	Presentation context addition list Presentation context deletion list Presentation context addition result list User data
P-ALTER-CONTEXT response P-ALTER-CONTEXT confirm	Presentation context addition result list Presentation context deletion result list User data
P-DATA request P-DATA indication	User data
P-TYPED-DATA request P-TYPED-DATA indication	User data
P-EXPEDITED-DATA request P-EXPEDITED-DATA indication	User data
P-CAPABILITY-DATA request P-CAPABILITY-DATA indication	User data
P-CAPABILITY-DATA response P-CAPABILITY-DATA confirm	User data
P-TOKEN-GIVE request P-TOKEN-GIVE indication	Tokens
P-TOKEN-PLEASE request P-TOKEN-PLEASE indication	Tokens User data
P-CONTROL-GIVE request P-CONTROL-GIVE indication	

TABLE 2/X.216 *(continued)*

SERVICE PRIMITIVE	PARAMETER
P-SYNC-MINOR request P-SYNC-MINOR indication	Type Synchronization point serial number User data
P-SYNC-MINOR response P-SYNC-MINOR confirm	Synchronization point serial number User data
P-SYNC-MAJOR request P-SYNC-MAJOR indication	Synchronization point serial number User data
P-SYNC-MAJOR response P-SYNC-MAJOR confirm	User data
P-RESYNCHRONIZE request	Resynchronize type Synchronization point serial number Tokens User data
P-RESYNCHRONIZE indication	Resynchronize type Synchronization point serial number Tokens Presentation context identification list User data
P-RESYNCHRONIZE response	Synchronization point serial number Tokens User data
P-RESYNCHRONIZE confirm	Synchronization point serial number Tokens Presentation context identification list User data
P-U-EXCEPTION-REPORT request P-U-EXCEPTION-REPORT indication	Reason User data
P-P-EXCEPTION-REPORT indication	Reason
P-ACTIVITY-START request P-ACTIVITY-START indication	Activity identifier User data

TABLE 2/X.216 *(continued)*

SERVICE PRIMITIVE	PARAMETER
P-ACTIVITY-RESUME request P-ACTIVITY-RESUME indication	Activity identifier Old activity identifier Synchronization point serial number Old session connection identifier User data
P-ACTIVITY-END request P-ACTIVITY-END indication	Synchronization point serial number User data
P-ACTIVITY-END response P-ACTIVITY-END confirm	User data
P-ACTIVITY-INTERRUPT request P-ACTIVITY-INTERRUPT indication	Reason
P-ACTIVITY-INTERRUPT response P-ACTIVITY-INTERRUPT confirm	
P-ACTIVITY-DISCARD request P-ACTIVITY-DISCARD indication	Reason
P-ACTIVITY-DISCARD response P-ACTIVITY-DISCARD confirm	

10.2 *P-CONNECT service*

This service is used to bring two identified PS-users into communication. Its successful use results in a presentation-connection, with an initial DCS, being established between them. This presentation-connection is available for their subsequent communication. This is a non-destructive service.

10.2.1 *Structure*

Note – There may be a session-service data size dependent limitation on this presentation service. This may prevent the delivery of the P-CONNECT indication and/or confirm service primitives.

The structure of the service primitives is shown in Table 3/X.216.

TABLE 3/X.216

P-CONNECT service primitive structure

Parameter name	Request	Indication	Response	Confirm
Calling-presentation-address	M	M		
Called-presentation-address	M	M		
Responding-presentation-address			M	M
Presentation context definition list	U	C(=)		
Presentation context definition result list		C	C	C(=)
Default context name	U	C(=)		
Default context result (Note)			C	C(=)
Quality of service	S	S	S	S
Presentation requirements	U	C	U	C(=)
Mode	M	M(=)		
Session requirements	S	S	S	S
Initial synchronization point serial number	S	S	S	S
Initial assignment of tokens	S	S	S	S
Session connection identifier	S	S	S	S
User data	U	C(=)	U	C(=)
Result (Note)			M	M(=)

M: presence of the parameter is mandatory;
U: presence of the parameter is a user option;
C: presence of the parameter is conditional;
S: parameter is as required by the session-service primitive which supports this service (see Recommendation X.215);
(=): when appended to one of the above, the value of the parameter is equal to the value of the parameter indicated in the column to the left;
blank: the parameter is not present.

Note – When the presentation-connection establishment request is rejected by the presentation-service-provider, the value of this parameter is provider-generated.

10.2.1.1 *Calling-presentation-address*

This is a presentation-address (see ISO 7498-3).

10.2.1.2 *Called-presentation-address*

This is a presentation-address (see ISO 7498-3).

10.2.1.3 *Responding-presentation-address*

This is a presentation-address (see ISO 7498-3).

10.2.1.4 *Presentation context definition list*

This parameter is present when the PS-user requires to place one or more presentation contexts in the DCS at the time of presentation-connection establishment. It consists of a list containing one or more items; each item contains two components, a presentation context identifiction and an abstract syntax name.

The presentation context identification components of this parameter exist to distinguish presentation contexts in communication between the PS-user and the local presentation-entity. The unambiguous identification of the presentation context to be established is required. The way this is achieved in a real open system is an implementation matter.

Note – A separate presentation context is associated with each abstract syntax name in the list of names in the Presentation context definition list parameter. If the same name occurs more than once, a separate and distinctly identified presentation context is generated for each occurrence.

10.2.1.5 *Presentation context definition result list*

This parameter indicates the acceptance or rejection of each of the presentation context definitions proposed in the Presentation context definition list parameter; it shall be present only if the Presentation context definition list parameter is present on the request and indication service primitives. The parameter takes the form of a list of result values; there is a one-to-one correspondence between these list elements and the contents of the presentation context definition list parameter. Each result value represents either "acceptance", "user-rejection", "user-rejection" or "provider-rejection". The values of the elements in this parameter are assigned by the presentation-service-provider on the indication service primitive and by the PS-user on the response service primitive.

When present in the indication service primitive, this parameter is used to identify to the responding PS-user, those proposed presentation context definitions which cannot be supported by the presentation-service-provider, by assigning the value "provider-rejection" to the appropriate list element. All other elements are assigned the value "acceptance", and the responding PS-user is restricted to modifying the value of only these accepted elements.

Values of this parameter in the response service primitive are delivered unchanged in the confirm service primitive.

10.2.1.6 *Default context name*

This parameter is present when the PS-user requires to identify explicitly the abstract syntax supported by the default context. It identifies an abstract syntax name.

10.2.1.7 *Default context result*

This parameter is provided by the responding PS-user or the presentation-service-provider. It indicates acceptance or rejection of a proposed default context, and is present if, and only if, the default context name parameter was present on the request and indication service primitives. On the response service primitive, it shall take the value "acceptance" or "user-rejection" as selected by the PS-user. On the confirm service primitive, it shall take the value from the response service primitive, or the value "provider-rejection" if the proposed default context is refused by the presentation-service-provider.

10.2.1.8 *Quality of Service*

This parameter provides the PS-user with access to the Quality of Service parameter of the session-service and is as described for that parameter in Recommendation X.215.

10.2.1.9 *Presentation requirements*

This parameter is present when the PS-user requires to select optional functional units of the presentation-service.

10.2.1.10 *Mode*

This parameter indicates the mode of operation of the Presentation Layer. It takes the value either "normal" or "X.410-1984". If the value is "normal", the mode of operation of the Presentation Layer is the normal mode. If the value is "X.410-1984", the mode of operation of the Presentation Layer is the X.410-1984 mode. In this mode of operation, the following restrictions apply:

a) the following parameters shall be absent in the P-CONNECT request service primitive: Presentation context definition list, Default context name, and Presentation requirements;

b) restrictions apply to the User data parameters of certain presentation-service primitives; these are listed in annex A.

10.2.1.11 *Session requirements*

This parameter provides the PS-user with access to the Session requirements parameter of the session-service and is as described for that parameter in Recommendation X.215.

10.2.1.12 *Initial synchronization point serial number*

This parameter provides the PS-user with access to the Initial synchronization point serial number parameter of the session-service and is as described for that parameter in Recommendation X.215.

10.2.1.13 *Initial assignment of tokens*

This parameter provides the PS-user with access to the Initial assignment of tokens parameter of the session-service and is as described for that parameter in Recommendation X.215.

10.2.1.14 *Session connection identifier*

This parameter provides the PS-user with access to the Session connection identifier parameter of the session-service and is as described for that parameter in Recommendation X.215.

10.2.1.15 *User data*

On all P-CONNECT service primitives, this parameter is one or more presentation data values (including any embedded presentation data values) from presentation contexts proposed in the Presentation context definition list parameter, if present; if the Presentation context definition list parameter is not present, then the User data parameter is one or more presentation data values from the proposed default context (either implicitly or explicitly defined in the P-CONNECT request).

10.2.1.16 *Result*

This parameter is provided by the responding PS-user or presentation-service-provider. It indicates the result of using the P-CONNECT service. The value of this parameter is one of:

a) "acceptance";

b) "user-rejection";

c) "provider-rejection".

The reasons for rejection of the presentation-connection are to be defined[3)].

10.2.2 *Connection procedure*

10.2.2.1 The presentation-service-provider conveys the Calling-presentation-address, Called-presentation-address, Mode, Initial synchronization point serial number, Initial assignment of tokens, Session connection identifier, and User data parameters unchanged from the initiating to the responding PS-user. The presentation-service-provider conveys the Responding-presentation-address, Initial synchronization point serial number, Initial assignment of tokens, Session connection identifier, and User data parameters unchanged from the responding to the initiating PS-user.

10.2.2.2 The connection characteristics specified by the Presentation requirements, Session requirements and Quality of Service parameters are subject to agreement between the PS-users and the presentation-service-provider. This agreement is achieved by a negotiation mechanism in which the presentation-service provider reserves the right to modify the values of these parameters specified in a request service primitive prior to their delivery in an indication service primitive. The values of these parameters in a response service primitive are delivered unchanged in a confirm service primitive and subject to the conditions below:

a) For the Presentation requirements and Session requirements parameters, the PS-user shall not select a functional unit in the response service primitive which was not selected in the indication service primitive.

b) The values of the Quality of Service parameter are subject to the negotiation rules specified in Recommendation X.215.

[3)] It is recognized that, with respect to reason values, work is still in progress to provide an integrated treatment across all the layers of the OSI Reference Model. As a consequence, an annex may be added to this Recommendation at a later time which reflects further developments and integration.

10.2.2.3 The Presentation context definition list parameter is optional in the P-CONNECT request service primitive; in its absence, the DCS is empty. When this parameter is present, the presentation contexts it specifies are available for use in the User data parameter; in its absence, only the default context is available for use.

If the Presentation context definition list parameter is present in the P-CONNECT request service primitive, then it shall also be present in the P-CONNECT indication service primitive if issued, together with the Presentation context definition result list parameter. In this case, the Presentation context definition result list parameter shall also be present in the P-CONNECT response and confirm service primitives.

10.2.2.4 The Default context name parameter is optional in the P-CONNECT request service primitive; if it is absent then the presentation-service-provider assumes that there is prior agreement on the definition of the default context. When present, this parameter specifies the abstract syntax supported by the default context.

If this parameter is present in the P-CONNECT request service primitive but cannot be supported by the presentation-service-provider, then no indication shall be issued and the initiating PS-user will receive a P-CONNECT confirm service primitive with a Default context result parameter value of "provider-rejection" and a Result parameter value of "provider-rejection".

If the presentation-service-provider supports the default context, then an indication shall be issued to the responding PS-user. If in the response and confirm service primitives, the Default context result parameter takes the value "User-rejection", then the Result parameter of these service primitives shall also take the value "user-rejection".

10.2.2.5 If any part of the User data parameter of the P-CONNECT request service primitive cannot be transferred to the responding PS-user, then no indication shall be issued and the initiating PS-user will receive a P-CONNECT confirm service primitive with a Result parameter value of "provider-rejection".

10.2.2.6 If the PS-user issues a P-CONNECT response service primitive with a Result parameter value of "acceptance", then the P-CONNECT confirm service primitive shall be issued with a Result parameter value of "acceptance" and the presentation-connection is established. If the PS-user issues a P-CONNECT response service primitive with a Result parameter value of "user-rejection", then the P-CONNECT confirm service primitive shall be issued with a Result parameter value of "user-rejection" together with any user data which was present on the response service primitive; the presentation-connection is not established. The responding PS-user shall not issue a P-CONNECT response service primitive with a Result parameter value of "acceptance" and a Default context result parameter value of "user-rejection".

10.2.2.7 If a P-CONNECT confirm service primitive is not acceptable to a PS-user, the PS-user may subsequently issue a P-U-ABORT request service primitive.

10.3 *P-U-ABORT service*

This service can be used by either PS-user to force the release of a presentation-connection at any time and have the peer PS-user informed of this termination. This service has effects which may not be sequenced with respect to preceding service invocations and its invocation is destructive.

10.3.1 *Structure*

The structure of the component service primitives is shown in Table 4/X.216.

10.3.1.1 *User data*

The presentation data values (including any embedded presentation data values) in this parameter are passed between PS-users and obey the rules of § 10.1; the interpretation of this data is an Application Layer matter. No other significance is attached to this data by the presentation-service. It can therefore be used for passing user reason information.

Note – If presentation data value is received from a proposed but not yet acknowledged presentation context, it is assumed that the P-U-ABORT overtook the acknowledgement. In this case the data is accepted and delivered as though the acknowledgement had been received.

TABLE 4/X.216

P-U-ABORT service

Parameter name	Request	Indication
User data	U	C(=)

U: presence of the parameter is a user option;
C: presence of the parameter is conditional;
(=): when appended to one of the above, the value of the parameter is equal to the value of the parameter indicated in the column to the left.

10.4 *P-P-ABORT service*

This service is the means by which the presentation-service-provider may indicate the termination of the presentation-connect for reasons internal to the presentation-service-provider. This service has effects which may not be sequenced with respect to preceding service invocations and its invocation is destructive.

10.4.1 *Structure*

The structure of the component service primitives is shown in Table 5/X.216.

TABLE 5/X.216

P-P-ABORT service

Parameter name	Indication
Provider reason	M

M: presence of the parameter is mandatory.

10.4.1.1 *Provider reason*

This parameter indicates the reason for the termination of the presentation-connection[4].

[4] It is recognized that, with respect to reason values, work is still in progress to provide an integrated treatment across all the layers of the OSI Reference Model. As a consequence, an annex may be added to this Recommendation at a later time which reflects further developments and integration.

10.5 *P-ALTER-CONTEXT service*

Note – This service is only available when the context management functional unit has been selected during presentation-connection establishment.

This service provides the following presentation context management facilities:

a) the creation of presentation contexts and their addition to the DCS;

b) the deletion of presentation contexts from the DCS.

This service has effects which are sequenced with respect to preceding service invocations and is non-destructive.

10.5.1 *Structure*

The structure of the component service primitives is shown in Table 6/X.216.

TABLE 6/X.216

P-ALTER-CONTEXT service

Parameter name	Request	Indication	Response	Confirm
Presentation context addition list	U	C(=)		
Presentation context deletion list	U	C(=)		
Presentation context addition result list		C	U	C(=)
Presentation context deletion result list			U	C(=)
User data	U	C(=)	U	C(=)

U: presence of the parameter is a user option;

C: presence of the parameter is conditional;

(=): when appended to one of the above, the value of the parameter is equal to the value of the parameter indicated in the column to the left;

blank:the parameter is not present.

10.5.1.1 *Presentation context addition list*

This parameter enables presentation context addition requirements to be specified. It takes the form of a list. Each item of the list represents a specification for a presentation context to be created and added to the DCS. An item contains two components, a presentation context identification and an abstract syntax name; these are both provided by the requestor of the service.

The presentation context identification components of this parameter exist to distinguish identification contexts in communication between the PS-user and the local identification-entry. The unambiguous identification of the identification context to be established is required. The way this is achieved in a real open system is an implementation matter.

Note – A separate presentation component is associated with each abstract syntax name in the list of names in the Presentation context addition list parameter. If the same name occurs more than once, or has been used in an earlier identification context addition, a separate and distinctly identified identification context is generated for each occurrence.

10.5.1.2 *Presentation context deletion list*

This parameter enables presentation context deletion requirements to be specified. It takes the form of a list. Each item in the list is the presentation context identification of a presentation context that is to be removed from the DCS.

10.5.1.3 *Presentation context addition result list*

This parameter indicates the acceptance or rejection of each of the presentation context additions proposed in the Presentation context addition list parameter; it shall be present only if the Presentation context addition list parameter is present on the request and indication service primitives. The parameter takes the form of a list of result values; there is a one-to-one order-preserving correspondence between these list elements and the contents of the presentation context addition list. Each result value represents either "acceptance", "user-rejection" or "provider-rejection". The values of the elements in this parameter are assigned by the presentation-service-provider on the indication service primitive and by the PS-user on the response service primitive.

When present in the indication service primitive, this parameter is used to identify to the accepting PS-user, those proposed presentation context additions which cannot be supported by the presentation-service-provider, by assigning the value "provider-rejection" to the appropriate list element. All other elements are assigned the value "acceptance", and the accepting PS-user is restricted to modifying the value of only these accepted elements.

Absence of this parameter is equivalent to acceptance of all proposed presentation context additions. Values of this parameter in the response service primitive are delivered unchanged in the confirm service primitive.

10.5.1.4 *Presentation context deletion result list*

This parameter indicates the acceptance or rejection of each of the presentation context deletions proposed in the Presentation context deletion list parameter; it shall be present only if the Presentation context deletion list parameter is present on the request and indication service primitives. The parameter takes the form of a list of result values; there is a one-to-one, order-preserving correspondence between these list elements and the contents of the presentation context deletion list. Each result value represents either acceptance or rejection by the PS-user.

Absence of this parameter is equivalent to acceptance of all proposed presentation context deletions. Values of this parameter in the response service primitive are delivered unchanged in the confirm service primitive.

10.5.1.5 *User data*

This parameter contains presentation data values (including any embedded presentation data values) from presentation contexts of the DCS, or from the default context if the DCS is empty. See § 10.5.2.

10.5.2 *Alter context procedure*

10.5.2.1 The accepted modifications to the DCS become effective:

a) for the acceptor when issuing the response service primitive;

b) for the requestor upon receiving the confirm service primitive.

A presentation context added to the DCS may be used for presentation data values of the User data parameter in the P-ALTER-CONTEXT response and confirm service primitives. A presentation context removed from the DCS may not be used for presentation data values of the User data parameter in the P-ALTER-CONTEXT response and confirm service primitives.

10.5.2.2 If the DCS is empty prior to invoking the P-ALTER-CONTEXT request service primitive, then the requestor shall use only the default context for the User data parameter. Moreover, while the P-ALTER-CONTEXT confirm service primitive is awaited, the requestor shall not issue presentation-service primitives containing User data parameters other than P-EXPEDITED, P-U-EXCEPTION-REPORT, P-RESYNCHRONIZE or P-U-ABORT.

10.5.2.3 If the DCS becomes empty as a result of the invocation of this service, then the acceptor shall use only the default context for the User data parameter of response and confirm service primitives.

10.6 *P-TYPED-DATA service*

This service provides the PS-user with access to the S-TYPED-DATA service of the Session Layer as described in the Session Service Definition (Recommendation X.215). This service has effects which are sequenced with respect to preceding service invocations and is non-destructive.

10.6.1 *Structure*

The structure of the component service primitives is shown in Table 7/X.216.

TABLE 7/X.216

P-TYPED-DATA service

Parameter name	Request	Indication
User data	M	M(=)

M: presence of the parameter is mandatory;

(=): when appended to one of the above, the value of the parameter is equal to the value of the parameter indicated in the column to the left.

10.6.1.1 *User data*

The presentation data values (including any embedded presentation data values) in this parameter are passed between PS-users and obey the rules of § 10.1; the interpretation of this data is an Application Layer matter. No other significance is attached to this data by the presentation-service.

10.7 *P-DATA service*

This service provides the PS-user with access to the S-DATA service of the Session Layer as described in the Session Service Definition (Recommendation X.215). This service has effects which are sequenced with respect to preceding service invocations and is non-destructive.

10.7.1 *Structure*

The structure of the component service primitives is shown in Table 8/X.216.

TABLE 8/X.216

P-DATA service

Parameter name	Request	Indication
User data	M	M(=)

M: presence of the parameter is mandatory;

(=): when appended to one of the above, the value of the parameter is equal to the value of the parameter indicated in the column to the left.

10.7.1.1 *User data*

The presentation data values (including any embedded presentation data values) in this parameter are passed between PS-users and obey the rules of § 10.1; the interpretation of this data is an Application Layer matter. No other significance is attached to this data by the presentation-service.

10.8 *P-RESYNCHRONIZE service*

This service provides the PS-user with access to the S-RESYNCHRONIZE session service as described in the Session Service Definition, Recommendation X.215. This service has effects which may not be sequenced with respect to preceding service invocations and is destructive.

10.8.1 *Structure*

The structure of the component service primitives is shown in Table 9//X.216.

TABLE 9/X.216

P-RESYNCHRONIZE service

Parameter name	Request	Indication	Response	Confirm
Resynchornization type	S	S		
Synchronization point serial number	S	S	S	S
Tokens	S	S	S	S
Presentation context identification list		C		C
User data	U	C(=)	U	C(=)

U: presence of the parameter is a user option;

C: presence of the parameter is conditional;

S: parameter is as required by the session-service primitive which supports this service (see Recommendation X.215);

(=): when appended to one of the above, the value of the parameter is equal to the value of the parameter indicated in the column to the left;

blank:the parameter is not present.

10.8.1.1 *Resynchronize type*

This parameter provides the PS-user with access to the Resynchronize type parameter of the resynchronize session service as described in Recommendation X.215.

10.8.1.2 *Synchronization point serial number*

This parameter provides the PS-user with access to the Synchronization point serial number of the resynchronize session service as described in Recommendation X.215.

10.8.1.3 *Tokens*

This parameter provides the PS-user with access to the Tokens parameter of the resynchronize session service as described in Recommendation X.215.

10.8.1.4 *Presentation context identification list*

This parameter consists of a list containing zero, one or more items; each item consists of a presentation context identification. This parameter is provided by the presentation-service-provider, see § 10.8.2.3.

10.8.1.5 *User data*

The presentation data values (including any embedded presentation data values) in this parameter are passed between PS-users and obey the rules of § 10.1; the interpretation of this data is an Application Layer matter. No other significance is attached to this data by the presentation-service.

10.8.2 *Resynchronization procedure*

10.8.2.1 The presentation-service-provider conveys the session defined parameters between the PS-users as specified by the session-service.

10.8.2.2 If the context management functional unit is not selected, then the Presentation context identification list parameter is not present. In this case, the contents of the DCS do not vary during the presentation connection.

10.8.2.3 If the context management functional unit is selected, then the Presentation context identification list parameter is present in the parameter P-RESYNCHRONIZE indication and confirm service primitives. This parameter lists all the presentation contexts that are members of the DCS.

The User data parameter in the P-SYNCHRONIZE service primitives contains presentation data values from presentation contexts which are members of the DCS at the invocation of the request or response service primitive respectively, but if a P-ALTER-CONTEXT confirm service primitive is awaited, then presentation contexts proposed for deletion may not be used.

10.8.2.4 If the context restoration functional unit is selected and the resynchronize type is either "restart" or "set", then the DCS may be restored when the request, indication, and confirm service primitives are invoked, according to the following rules:

a) If the specified synchronization point serial number is less than or equal to the lowest synchronization point serial number that has been used on the presentation-connection and has not been specified in a P-SYNC-MAJOR or P-SYNC-MINOR request or indication service primitive on the current presentation-connection, then the DCS is restored to that immediately after presentation-connection establishment;

b) If the specified synchronization point serial number minus one has been specified in P-SYNC-MINOR or P-SYNC-MAJOR request or indication service primitive on the current presentation-connection, then the DCS is restored to that which was current at the invocation of the P-SYNC-MINOR or P-SYNC-MAJOR service;

c) If the specified synchronization point serial number is greater than the current synchronization point serial number for either of the PS-users or greater than the lowest synchronization point serial number used on the presentation-connection but is not known to one of the presentation-entities, then the resulting DCS is unchanged.

On completion of this, any previous P-SYNC-MINOR or P-SYNC-MAJOR specifying greater synchronization point serial numbers are disregarded in evaluating future P-RESYNCHRONIZE and P-ACTIVITY-RESUME procedures.

If the activity management functional unit has been selected for use on the presentation-connection, then only the P-SYNC-MAJOR and P-SYNC-MINOR service primitives within the current activity are taken into account. (See also § 10.22.2.)

10.9 *P-ACTIVITY-START service*

This service provides the PS-user with access to the S-ACTIVITY-START session service as described in Recommendation X.215. This service has effects which are sequenced with respect to preceding service invocations and is non-destructive.

10.9.1 *Structure*

The structure of the component service primitives is shown in Table 10/X.216.

TABLE 10/X.216

P-ACTIVITY-START service

Parameter name	Request	Indication
Activity identifier	S	S
User data	U	C(=)

U: presence of the parameter is a user option;

C: presence of the parameter is conditional;

S: parameter is as required by the session-service primitive which supports this service (see Recommendation X.215);

(=): when appended to one of the above, the value of the parameter is equal to the value of the parameter indicated in the column to the left.

10.9.1.1 *Activity identifier*

If the context restoration functional unit is selected, this parameter shall uniquely identify the activity within the set of previously interrupted activities.

Note – If either the PS-user can resume an interrupted activity, the Activity identifier parameter value should be different from the Activity identifier parameter values of all interrupted activities which were started by this PS-user.

10.9.1.2 *User data*

The presentation data values (including any embedded presentation data values) in this parameter are passed between PS-users and obey the rules of § 10.1; the interpretation of this data is an Application Layer matter. No other significance is attached to this data by the presentation-service.

10.10 *P-ACTIVITY-RESUME service*

This service provides the PS-user with access to the S-ACTIVITY-RESUME session service as described in Recommendation X.215. This service has effects which are sequenced with respect to preceding service invocations and is non-destructive.

10.10.1 *Structure*

The structure of the component service primitives is shown in Table 11/X.216.

10.10.1.1 *Activity identifier*

This parameter provides the PS-user with access to the Activity identifier parameter of the session activity resume service as described in Recommendation X.215.

10.10.1.2 *Old activity identifier*

This parameter provides the PS-user with access to the Old activity identifier parameter of the session activity resume service as described in Recommendation X.215. This parameter shall uniquely identify the activity within the set of interrupted activities.

TABLE 11/X.216

P-ACTIVITY-RESUME service

Parameter name	Request	Indication
Activity identifier	S	S
Old activity identifier	S	S
Synchronization point serial number	S	S
Old session connection identifier	S	S
User data	U	C(=)

U: presence of the parameter is a user option;

C: presence of the parameter is conditional;

S: parameter is as required by the session-service primitive which supports this service (see Recommendation X.215);

(=): when appended to one of the above, the value of the parameter is equal to the value of the parameter indicated in the column to the left.

10.10.1.3 *Synchronization point serial number*

This parameter provides the PS-user with access to the Synchronization point serial number parameter of the session activity resume service as described in Recommendation X.215.

10.10.1.4 *Old session connection identifier*

This parameter provides the PS-user with access to the Old session connection identifier parameter of the session activity resume service as described in Recommendation X.215.

10.10.1.5 *User data*

The presentation data values (including any embedded presentation data values) in this parameter are passed between PS-users and obey the rules of § 10.1; the interpretation of this data is an Application Layer matter. No other significance is attached to this data by the presentation-service.

10.10.2 *Activity resume procedure*

10.10.2.1 The presentation-service-provider conveys the session defined parameters between the PS-users as specified by the session-service.

10.10.2.2 If the context restoration functional unit is not selected, then the DCS is unchanged.

10.10.2.3 If the context restoration functional unit is selected, then the DCS is specified as follows:

a) If the Old activity identifier parameter is equal to the Activity identifier parameter of an activity interrupted within the presentation-connection, then the DCS is restored to the one at the time that the value of the Synchronization point serial number parameter was specified in a S-SYNC-MINOR or S-SYNC-MAJOR service within the activity.

b) If the value of the Synchronization point serial number parameter had not been so specified within the activity in this presentation-connection, then the DCS is unchanged.

On completion of this, any previously invoked P-SYNC-MINOR or P-SYNC-MAJOR service, specifying greater synchronization point serial numbers are disregarded in evaluating future P-RESYNCHRONIZE and P-ACTIVITY-RESUME procedures.

Note – When the context restoration functional unit is selected, use of this non-confirmed-service without protection against crossing with P-DATA or P-TYPED-DATA services may result in a P-P-ABORT due to unreadable User-data. Such collisions may be avoided by strict separation of data exchanged outside of activity from that exchanged within an activity.

10.11 *P-ACTIVITY-INTERRUPT service*

This service provides the PS-user with access to the S-ACTIVITY-INTERRUPT session service as described in Recommendation X.215. This service has effects which may not be sequenced with respect to preceding service invocations and is destructive.

10.11.1 *Structure*

The structure of the component service primitives is shown in Table 12/X.216.

TABLE 12/X.216

P-ACTIVITY-INTERRUPT service

Parameter name	Request	Indication	Response	Confirm
Reason	S	S		

S: parameter is as required by the session-service primitive which supports this service (see Recommendation X.215);
blank:the parameter is not present.

10.11.1.1 *Reason*

This parameter provides the PS-user with access to the Reason parameter of the session activity interrupt service as described in Recommendation X.215.

10.11.2 *Activity interrupt procedure*

10.11.2.1 If the context restoration functional unit is not selected, then no action is taken on the DCS.

10.11.2.2 If the context restoration functional unit is selected, then the DCS is aligned with the inter-activity DCS on the issuing of the response and confirm service primitives for this service.

10.11.2.3 Any P-ACTIVITY-INTERRUPT service primitive issued outside of an activity shall have no effect on the DCS.

10.12 *P-ACTIVITY-DISCARD service*

This service provides the PS-user with access to the S-ACTIVITY-DISCARD session service as described in Recommendation X.215. This service has effects which may not be sequenced with respect to preceding service invocations and is destructive.

10.12.1 *Structure*

The structure of the component service primitives is shown in Table 13/X.216.

TABLE 13/X.216

P-ACTIVITY-DISCARD service

Parameter name	Request	Indication	Response	Confirm
Reason	S	S		

S: parameter is as required by the session-service primitive which supports this service (see Recommendation X.215);
blank:the parameter is not present.

10.12.1.1 *Reason*

This parameter provides the PS-user with access to the Reason parameter of the session activity discard service as described in Recommendation X.215.

10.12.2 *Activity discard procedure*

10.12.2.1 If the context restoration functional unit is not selected, then no action is taken on the DCS.

10.12.2.2 If the context restoration functional unit is selected, then the DCS is aligned with the inter-activity DCS on the issuing of the response and confirm service primitives for this service.

10.13 *P-ACTIVITY-END service*

This service provides the PS-user with access to the S-ACTIVITY-END session service as described in Recommendation X.215. This service has effects which are sequenced with respect to preceding service invocations and is non-destructive.

10.13.1 *Structure*

The structure of the component service primitives is shown in Table 14/X.216.

10.13.1.1 *Synchronization point serial number*

This parameter provides the PS-user with access to the Synchronization point serial number parameter of the session activity end service as described in Recommendation X.215.

10.13.1.2 *User data*

The presentation data values (including any embedded presentation data values) in this parameter are passed between PS-users and obey the rules of § 10.1; the interpretation of this data is an Application Layer matter. No other significance is attached to this data by the presentation-service.

TABLE 14/X.216

P-ACTIVITY-END service

Parameter name	Request	Indication	Response	Confirm
Synchronization point serial number	S	S		
User data	U	C(=)	U	C(=)

U: presence of the parameter is a user option;

C: presence of the parameter is conditional;

S: parameter is as required by the session-service primitive which supports this service (see Recommendation X.215);

(=): when appended to one of the above, the value of the parameter is equal to the value of the parameter indicated in the column to the left;

blank:the parameter is not present.

10.13.2 *Activity-End procedure*

10.13.2.1 The presentation-service-provider conveys the session defined parameters between the PS-users as specified by Recommendation X.215.

10.13.2.2 If the context restoration functional unit is not selected, then no action is taken on the DCS.

10.13.2.3 If the context restoration functional unit is selected, then the DCS is aligned with the inter-activity DCS on the issuing of the response and confirm service primitives for this service.

10.14 *P-CAPABILITY-DATA service*

This service provides the PS-user with access to the S-CAPABILITY-DATA service of the Session Layer as described in Recommendation X.215. This service has effects which are sequenced with respect to preceding service invocations and is non-destructive.

10.14.1 *Structure*

The structure of the component service primitives is shown in Table 15/X.216.

TABLE 15/X.216

P-CAPABITILITY-DATA service

Parameter name	Request	Indication	Response	Confirm
User data	U	C(=)	U	C(=)

U: presence of the parameter is a user option;

C: presence of the parameter is conditional;

(=): when appended to one of the above, the value of the parameter is equal to the value of the parameter indicated in the column to the left.

10.14.1.1 *User data*

The parameter data values (including any embedded presentation data values) in this parameter are passed between PS-users and obey the rules of § 10.1; the interpretation of this data is an Application Layer matter. No other significance is attached to this data by the presentation-service.

10.15 *P-CONTROL-GIVE service*

This service provides the PS-user with access to the S-CONTROL-GIVE service of the Session Layer as described in Recommendation X.215. This service has effects which are sequenced with respect to preceding service invocations and is non-destructive.

10.15.1 *Structure*

This service has no parameters.

10.16 *P-TOKEN-GIVE service*

This service provides the PS-user with access to the S-TOKEN-GIVE service of the Session Layer as described in Recommendation X.215. This service has effects which are sequenced with respect to preceding service indications and is non-destructive.

10.16.1 *Structure*

The structure of the component service primitives is shown in Table 16/X.216.

TABLE 16/X.216

P-TOKEN-GIVE service

Parameter name	Request	Indication
Tokens	S	S

S: parameter is as required by the session-service primitive which supports this service (see Recommendation X.215).

10.16.1.1 *Tokens*

This parameter corresponds to the Tokens parameter of the session-service, see Recommendation X.215.

10.17 *P-TOKEN-PLEASE service*

This service provides the PS-user with access to the S-TOKEN-PLEASE service of the Session Layer as described in Recommendation X.215. This service has effects which are sequenced with respect to preceding service invocations and is non-destructive.

10.17.1 *Structure*

The structure of the component service primitives is shown in Table 17/X.216.

TABLE 17/X.216

P-TOKEN-PLEASE service

Parameter name	Request	Indication
Tokens	S	S
User data	U	C(=)

U: presence of the parameter is a user option;

C: presence of the parameter is conditional;

S: parameter is as required by the session-service primitive which supports this service (see Recommendation X.215);

(=): when appended to one of the above, the value of the parameter is equal to the value of the parameter indicated in the column to the left.

10.17.1.1 *Tokens*

This parameter corresponds to the Tokens parameter of the session-service, see Recommendation X.215.

10.17.1.2 *User data*

The presentation data values (including any embedded presentation data values) in this parameter are passed between PS-users and obey the rules of § 10.1; the interpretation of this data is an Application Layer matter. No other significance is attached to this data by the presentation-service.

10.18 *P-U-EXCEPTION-REPORT service*

This service provides the PS-user with access to the S-U-EXCEPTION-REPORT service of the Session Layer as described in Recommendation X.215. This service has effects which are sequenced with respect to preceding service invocations and is destructive.

10.18.1 *Structure*

The structure of the component service primitives is shown in Table 18/X.216.

10.18.1.1 *Reason*

This parameter corresponds to the Reason parameter of the session-service, see Recommendation X.215.

10.18.1.2 *User data*

The presentation data values (including any embedded presentation data values) in this parameter are passed between PS-users and obey the rules of § 10.1; the interpretation of this data is an Application Layer matter. No other significance is attached to this data by the presentation-service.

TABLE 18/X.216

P-U-EXCEPTION-REPORT service

Parameter name	Request	Indication
Reason	S	S
User data	U	C(=)

U: presence of the parameter is a user option;

C: presence of the parameter is conditional;

S: parameter is as required by the session-service primitive which supports this service (see Recommendation X.215);

(=): when appended to one of the above, the value of the parameter is equal to the value of the parameter indicated in the column to the left.

10.19 *P-P-EXCEPTION-REPORT service*

This service gives the PS-user visibility to the S-P-EXCEPTION-REPORT service of the Session Layer as described in Recommendation X.215. This service is destructive.

10.19.1 *Structure*

The structure of the component service primitives is shown in Table 19/X.216.

TABLE 19/X.216

P-P-EXCEPTION-REPORT service

Parameter name	Indication
Reason	S

S: parameter is as required by the session-service primitive which supports this service (see Recommendation X.215).

10.19.1.1 *Reason*

This parameter corresponds to the Reason parameter of the session-service, see Recommendation X.215.

10.20 *P-EXPEDITED-DATA service*

This service provides the PS-user with access to the S-EXPEDITED-DATA service of the Session Layer as described in Recommendation X.215. This service has effects which may not be sequenced with respect to preceding service invocations and is non-destructive.

10.20.1 *Structure*

The sructure of the component service primitives is shown in Table 20/X.216.

TABLE 20/X.216

P-EXPEDITED-DATA service

Parameter name	Request	Indication
User data	M	M (=)

M: presence of the parameter is mandatory;

(=): when appended to one of the above, the value of the parameter is equal to the value of the parameter indicated in the column to the left.

10.20.1.1 *User data*

The presentation data values (including any embedded presentation data values) in this parameter are passed between PS-users and are from the default context; the interpretation of this data is an Application Layer matter. No other significance is attached to this data by the presentation-service.

10.21 *P-SYNC-MINOR service*

This service provides the PS-user with access to the S-SYNC-MINOR service of the Session Layer as described in Recommendation X.215. This service has effects which are sequenced with respect to preceding service invocations and is non-destructive.

10.21.1 *Structure*

The structure of the component service primitives is shown in Table 21/X.216.

10.21.1.1 *Type*

This parameter corresponds to the Type parameter of the session service, see Recommendation X.215.

10.21.1.2 *Synchronization point serial number*

This parameter corresponds to the Synchronization point serial number parameter of the session-service, see Recommendation X.215.

10.21.1.3 *User data*

The presentation data values (including any embedded presentation data values) in this parameter are passed between PS-users and obey the rules of § 10.1; the interpretation of this data is an Application Layer matter. No other significance is attached to this data by the presentation-service.

TABLE 21/X.216

P-SYNC-MINOR service

Parameter name	Request	Indication	Response	Confirmation
Type	S	S		
Synchronization point serial number	S	S	S	S
User data	U	C(=)	U	C(=)

U: presence of the parameter is a user option;

C: presence of the parameter is conditional;

S: parameter is as required by the session-service primitive which supports this service (see Recommendation X.215);

(=): when appended to one of the above, the value of the parameter is equal to the value of the parameter indicated in the column to the left;

blank: the parameter is not present.

10.22 *P-SYNC-MAJOR service*

This service provides the PS-user with access to the S-SYNC-MAJOR service of the Session Layer as described in Recommendation X.215. This service has effects which are sequenced with respect to preceding service invocations and is non-destructive.

10.22.1 *Structure*

The structure of the component service primitives is shown in Table 22/X.216.

TABLE 22/X.216

P-SYNC-MAJOR service

Parameter name	Request	Indication	Response	Confirmation
Synchronization point serial number	S	S		
User data	U	C(=)	U	C(=)

U: presence of the parameter is a user option;

C: presence of the parameter is conditional;

S: parameter is as required by the session-service primitive which supports this service (see Recommendation X.215);

(=): when appended to one of the above, the value of the parameter is equal to the value of the parameter indicated in the column to the left;

blank: the parameter is not present.

10.22.1.1 *Synchronization point serial number*

This parameter corresponds to the Synchronization point serial number parameter of the session-service, see Recommendation X.215.

10.22.1.2 *User data*

The presentation data values (including any embedded presentation data values) in this parameter are passed between PS-users and obey the rules of § 10.1; the interpretation of this data is an Application Layer matter. No other signficance is attached to this data by the presentation-service.

10.22.2 *Major-synchronize procedure*

Any previous P-SYN-MINOR and P-SYNC-MAJOR shall be disregarded in evaluating future P-RESYNCHRONIZE and/or P-ACTIVITY-RESUME procedures.

10.23 *P-RELEASE service*

This service provides the PS-user with access to the S-RELEASE service of the Session Layer as described in Recommendation X.215. This service has effects which are sequenced with respect to preceding serivce invocations and is non-destructive.

This service is also used to terminate the presentation-connection in an orderly way.

10.23.1 *Structure*

The structure of the component service primitives is shown in Table 23/X.216.

TABLE 23/X.216

P-RELEASE service

Parameter name	Request	Indication	Response	Confirmation
Result			S	S
User data	U	C(=)	U	C(=)

U: presence of the parameter is a user option;

C: presence of the parameter is conditional;

S: parameter is as required by the session-service primitive which supports this service (see Recommendation X.215);

(=): when appended to one of the above, the value of the parameter is equal to the value of the parameter indicated in the column to the left;

blank: the parameter is not present.

10.23.1.1 *Result*

This parameter corresponds to the Result parameter of the session-service, see Recommendation X.215.

10.23.1.2 *User data*

The presentation data values (including any embedded presentation data values) in this parameter are passed between PS-users and obey the rules of § 10.1; the interpretation of this data is an Application Layer matter. No other significance is attached to this data by the presentation-service.

10.23.2 *Release procedure*

The presentation-connection is released when the session-connection is released, as described in Recommendation X.215.

Note – The procedures governing the behaviour of the P-RELEASE request, indication, response, and confirm presentation-service primitives correspond to those governing the behaviour described in Recommendation X.215 for the S-RELEASE request, indication, response and confirm session-service primitives, respectively.

11 Sequences

This section defines the interrelationships among the facilities and the services of the Presentation Layer.

It specifies for a service (or a group of "similar" services) under which conditions it/they may not be invoked at a particular PCEP, which service procedures are disrupted by the invocation of this/these service and which service invocations will disrupt the procedures of this/these service.

In addition, the following general rules apply:

a) Services may only be invoked if the corresponding functional unit has been selected during presentation-connection establishment. The services of the (session and presentation) kernel functional units are always available.

b) A service invocation is independent of any token unless it is specified otherwise in this section.

Implicitly all sequencing rules of the session-service will apply, i.e. it is dependent on the mapping to the session-service which additional sequencing rules will apply. This Recommendation only specifies those sequencing rules which are not already determined by the session-service.

Note – In particular, this implies that the P-ALTER-CONTEXT, P-TYPED-DATA and P-DATA request service primitives should not be invoked if a P-SYNC-MAJOR, P-RESYNCHRONIZE, P-ACTIVITY-INTERRUPT, P-ACTIVITY-END, P-ACTIVITY-DISCARD or P-RELEASE confirm service primitive is awaited.

All sequences of service invocations which are not explicitly prohibited by this section (and which are not prohibited by the session-service) are permitted and need not be explicitly specified in this section.

Note – The mapping of the presentation-service to the session-service is given in Recommendation X.226. The session-service (see Recommendation X.215) imposes sequencing rules which prevent the invocation of the P-ALTER-CONTEXT request or response service primitive while a P-SYNC-MAJOR, P-ACTIVITY-END, P-CAPABILITY-DATA or P-RELEASE confirm service primitive is awaited. Therefore, to avoid deadlock, the requestor of a P-ALTER-CONTEXT request service primitive should respond to a P-SYNC-MAJOR, P-ACTIVITY-END, P-CAPABILITY-DATA or P-RELEASE indicaiton service primitive without awaiting the P-ALTER-CONTEXT confirm service primitive.

11.1 *P-CONNECT service*

11.1.1 *Type of service*

This is a confirmed-service.

11.1.2 *Invocation restrictions*

This service cannot be invoked on an established presentation-connection.

11.1.3 *Disrupted service procedures*

This service does not disrupt any presentation-service procedures.

11.1.4 *Disrupting services*

The procedure of this service can be disrupted by the P-U-ABORT service or the P-P-ABORT service.

11.1.5 *Other sequencing information*

Simultaneous attempts by both PS-users to establish a presentation-connection are treated independently by the presentation-service-provider. Dependent on the actions of the PS-users, this may result in zero, one or two presentation-connections being established.

11.2 *P-U-ABORT service*

11.2.1 *Type of service*

This is a non-confirmed-service.

11.2.2 *Invocation restrictions*

This service can be invoked at any time by either PS-user.

11.2.3 *Disrupted service procedures*

This service disrupts all presentation-service procedures. In a collision of the P-P-ABORT service with the invocation of the P-U-ABORT service at one PCEP, the P-P-ABORT indication service primitive is only invoked at the peer PCEP.

In case of a collision between two invocations of the P-U-ABORT service, neither indication service primitive is delivered since the presentation-connection is already terminated at both ends.

11.2.4 *Disrupting services*

In case of a collision between two invocations of the P-U-ABORT service neither indication service primitive is delivered since the presentation-connection is already terminated at both ends.

In case of a collision of the P-P-ABORT service with the invocation of the P-U-ABORT service, the P-U-ABORT service procedure is disrupted.

11.3 *P-P-ABORT service*

11.3.1 *Type of service*

This is a provider-initiated-service.

11.3.2 *Invocation restrictions*

This service can be invoked at any time by the presentation-service-provider.

11.3.3 *Disrupted service procedures*

This service disrupts all presentation-service procedures.

11.3.4 *Disrupting services*

In case of a collision of the P-P-ABORT service with the invocation of the P-U-ABORT service at one PCEP, the P-P-ABORT indication service primitive is only invoked at the peer PCEP.

11.4 *P-ALTER-CONTEXT service*

11.4.1 *Type of service*

This is a confirmed-service.

11.4.2 *Invocation restrictions*

These services may only be invoked on an established presentation-connection.

A P-ALTER-CONTEXT request service primitive shall not be invoked while awaiting a P-ALTER-CONTEXT confirm service primitive.

11.4.3 *Disrupted service procedures*

No presentation-service procedure is disrupted by this service.

11.4.4 *Disrupting services*

The procedure of this service can be disrupted by the P-U-ABORT, P-P-ABORT, P-U-EXCEPTION-REPORT, P-P-EXCEPTION-REPORT, P-RESYNCHRONIZE, P-ACTIVITY-INTERRUPT and P-ACTIVITY-DISCARD services.

If the presentation-connection is released, the P-ALTER-CONTEXT service procedure is disrupted.

11.4.5 *Other sequencing restrictions*

The following collisions of these services may occur:

P-ALTER-CONTEXT/P-ALTER-CONTEXT

These colliding services are treated independently by the presentation-service-provider.

11.5 *P-TYPED-DATA and P-DATA services*

11.5.1 *Type of service*

These are non-confirmed-services.

11.5.2 *Invocation restrictions*

These services may only be invoked on an established presentation-connection.

The P-DATA service may be subject to data token control.

11.5.3 *Disrupted service procedures*

No presentation-service procedure is disrupted by these services.

11.5.4 *Disrupting services*

The procedure of these services may be disrupted by the P-U-ABORT, P-P-ABORT, P-U-EXCEPTION-REPORT, P-RESYNCHRONIZE, P-P-EXCEPTION-REPORT, P-ACTIVITY-INTERRUPT and P-ACTIVITY-DISCARD services.

11.5.5 *Context-dependent restrictions*

If the DCS is empty when P-ALTER-CONTEXT confirm service primitive is awaited, then these services shall not be invoked.

11.6 *P-CAPABILITY-DATA service*

11.6.1 *Type of service*

This is a confirmed-service.

11.6.2 *Invocation restrictions*

This service may only be invoked on an established presentation-connection.

The P-CAPABILITY-DATA service is subject to token control imposed by the session-service.

11.6.3 *Disrupted service procedures*

No presentation-service procedure is disrupted by this service.

11.6.4 *Disrupting services*

The procedure of this service may be disrupted by the P-U-ABORT, P-P-ABORT, P-U-EXCEPTION-REPORT, P-P-EXCEPTION-REPORT, and P-ACTIVITY-INTERRUPT services.

11.6.5 *Context-dependent restrictions*

If the DCS is empty when P-ALTER-CONTEXT confirm service primitive is awaited, then this service shall not be invoked.

11.7 *P-EXPEDITED-DATA service*

11.7.1 *Type of service*

This service is non-confirmed.

11.7.2 *Invocation restrictions*

No invocation restrictions beyond those imposed on this service by the session-service (Recommendation X.215).

11.7.3 *Disrupted service procedures*

No presentation-service procedure is disrupted by this service.

11.7.4 *Disrupting services*

There are no sequencing rules in addition to those described by the session-service.

11.8 *P-SYNC-MINOR, P-SYNC-MAJOR, P-RELEASE, P-ACTIVITY-START, P-PLEASE-TOKENS, P-GIVE-TOKENS, P-GIVE-CONTROL, P-ACTIVITY-END and P-ACTIVITY-RESUME services*

11.8.1 *Type of services*

The types of these services are descrribed in Recommendation X.215.

11.8.2 *Invocation restrictions*

In addition to the invocation restrictions imposed on these services by the session-service, when the context restoration functional unit has been selected, the following invocation restrictions apply:

P-SYNC-MINOR, P-SYNC-MAJOR, P-ACTIVITY-START, P-ACTIVITY-END and P-ACTIVITY-RESUME request service primitives shall not be invoked if a P-ALTER-CONTEXT confirm service primitive is awaited.

11.8.3 *Disrupted service procedures*

There are no sequencing rules in addition to those imposed by the session-service.

11.8.4 *Disrupting services*

There are no sequencing rules in addition to those described by the session-service.

11.9 *P-RESYNCHRONIZE, P-U-EXCEPTION-REPORT, P-P-EXCEPTION-REPORT, P-ACTIVITY-INTERRUPT, AND P-ACTIVITY-DISCARD services*

11.9.1 *Type of services*

The types of these services are descrribed in Recommendation X.215.

11.9.2 *Disrupted service procedures*

In addition to the sequencing rules described by the session-service, these services may disrupt the procedures of the context management and information transfer facilities of the presentation-service.

11.9.3 *Disrupting services*

There are no sequencing rules in addition to those described by the session-service.

ANNEX A

(to Recommendation X.216)

Restrictions on the Use of the Presentation-service in X.410-1984 Mode

The use of the X.410-1984 mode of operation of the Presentation Layer imposes some restrictions on the abstract syntax of presentation data values used in the User data parameters of certain presentation-service primitives.

A.1 *P-CONNECT service*

The User data parameters of these presentation-service primitives are restricted to a single presentation data value of ASN.1 type SET.

A.2 *P-U-ABORT service*

The User data parameters of these presentation-service primitives are restricted to a single presentation data value of ASN.1 type SET.

A.3 *P-TOKEN-PLEASE service*

The User data parameters of these presentation-service primitives are restricted to a single presentation data value of ASN.1 type INTEGER.

A.4 *P-DATA service*

The User data parameters of these presentation-service primitives are restricted to a single presentation data value of ASN.1 type OCTET STRING.

Note – The OCTET STRING value may have been produced by local application of the syntax-matching service (see Recommendation X.200, § 7.2.4.1) to a value of some other type. Such application is outside of the scope of this Recommendation.

ASSOCIATION CONTRO SERVICE DEFINITION 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.210 defines the Open System Interconnection (OSI) Layer Service Definition Conventions;

(c) that Recommendation X.215 defines the Session Service Definition for Open Systems Interconnection for CCITT Applications;

(d) that Recommendation X.216 defines the Presentation Service Definition of Open Systems Interconnection for CCITT Applications;

(e) that Recommendation X.220 specifies the use of X.200 series protocols in CCITT Applications;

(f) that Recommendation X.227 specifies the Association Control Protocol Specification 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; and

(h) that there is a need for common Association Control to support various applications,

unanimously declares

that this Recommendation defines the Association Control Service of Open Systems Interconnection for CCITT Applications as given in the Scope and Field of Application.

CONTENTS

[1] Recommendation X.217 and ISO 8649 [Information processing systems – Open Systems Interconnection – Service definition 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 inter-connection recommendations, the interconnection of information processing systems:

- from different maufacturers;
- under different managements;
- of different levels of complexity; and
- of different technologies.

0.3 This Recommendation recognizes that application-processes may wish to communicate with each other for a wide variety of reasons. However, any communication will require the performance of certain services independent of the reasons for communication. The application-service-element defined herein provides such services.

0.4 This Recommendation defines services provided by the application service element for application-association control: the Association Control Service Element (ACSE). The ACSE provides basic facilities for the control of an application-association between two application-entities which communicate by means of a presentation-connection.

0.5 The use of services defined in this Recommendation is also governed by the use of the presentation-service (X.216) and the session-service (X.215).

0.6 It is recognized that, with respect to ACSE Quality of Services (QOS), described in § 9, work is still in progress to provide an integrated treatment of QOS across all 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

This Recommendation defines ACSE services for application-association control in an open systems interconnection environment. The ACSE services are provided by the use of the ACSE protocol (X.227) in conjunction with the presentation-service (X.216). The ACSE services assume as a minimum the use of the presentation-service Kernel functional unit.

This Recommendation does not specify individual implementations or products nor does it constrain the implementation of entities and interfaces within a computer system.

No requirement is made for conformance to this Recommendation.

2 References

Recommendation X.200	Reference Model of Open Systems Interconnection for CCITT applications. (See also ISO 7498-1)
Recommendation X.210	OSI layer service definition conventions. (See also ISO TR8509)
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.225	Session protocol specification for Open Systems Interconnection for CCITT applications. (See also ISO 8327 and ISO 8327 Addendum 2).
Recommendation X.227	Association Control protocol specification for Open Systems Interconnection for CCITT applications. (See also ISO 8650).
Recommendation X.410-1984	CCITT Recommendation X.410: Message Handling Systems: Remote Operation and Reliable Transfer Server.
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-protocol;
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).

3.4 *Presentation service definitions*

This Recommendation makes use of the following terms defined in Recommendation 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*

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

3.5.1 **application-association; association**

A cooperative relationship between two application-entities, formed by their exchange of application-protocol-control-information through their use of presentation-services.

[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 provides for the transfer of an application-entity title value by the transfer of its component values.

3.5.2 **application context**

An explicitly identified set of application-service-elements, related options and any other necessary information for the interworking of application-entities on an application-association.

Note – This definition is subject to refinement as a result of ongoing work in the area of the Application Layer structure.

3.5.3 **Association Control Service Element**

The particular application-service-element defined in this Recommendation.

3.5.4 **ACSE service-user**

The part of the application-entity which makes use of ACSE services.

3.5.5 **ACSE service-provider**

An abstraction of the totality of those entities which provide ACSE services to peer ACSE service-users.

3.5.6 **requestor**

The ACSE service-user which issues the request primitive for a particular ACSE service. For a confirmed service, it also receives the confirm primitive.

3.5.7 **acceptor**

The ACSE service-user which receives the indication primitive for a particular ACSE service. For a confirmed service, it also issues the response primitive.

3.5.8 **association-initiator**

The ACSE service-user which initiates a particular association, i.e. the requestor of the A-ASSOCIATE service which establishes the association.

3.5.9 **association-responder**

The ACSE service-user which is not the initiator of a particular association, i.e. the acceptor of the A-ASSOCIATE service which establishes the association.

3.5.10 **normal mode**

The mode of ACSE operation which results in the transfer of ACSE semantics, using the presentation-service.

3.5.11 **X.410-1984 mode**

The mode of ACSE operation which allows ACSE service-users to interwork using the protocol specified in CCITT Recommendation X.410-1984. The use of this mode results in no transfer of ACSE semantics.

3.5.12 **disrupt**

A service procedure is disrupted by another service procedure if the second service results in service primitives not being used as specified for the procedure of the first service.

4 Abbreviations

The following abbreviations are used in this Recommendation.

ACSE	Association Control Service Element
AE	application-entity

ASE	application-service-element
OSI	Open Systems Interconnection
QOS	Quality of Service

5 Conventions

5.1 This Recommendation defines services for the ACSE following the descriptive conventions defined in Recommendation X.210. In § 9, the definition of each ACSE service includes a table which lists the parameters of its primitives. For a given primitive, the presence of each parameter is described by one of the following values.

blank	not applicable
C	conditional
M	mandatory
P	subject to conditions defined in X.216
U	user option

5.2 In addition, the notation (=) indicates that a parameter value is semantically equal to the value to its left in the table.

6 Basic concepts

6.1 The reference model (X.200) represents communication between a pair of application-processes (APs) in terms of communication between their application-entities (AEs) using the presentation-service. The functionality of an AE is factored into a number of application-service-elements (ASEs). The interaction between AEs is described in terms of the use of their ASEs' services.

6.2 This Recommendation introduces the additional modelling concepts of application-association and application context.

6.3 An application-association is a cooperative relationship between two AEs. It provides the necessary frame of reference between the AEs in order that they may interwork effectively. This relationship is formed by the exchange of application-protocol-control-information between the application-entities through their use of presentation-services.

6.4 An application context is an explicitly identified set of application-service-elements, related options and any other necessary information for the interworking of application-entities on an application association.

7 Service overview

7.1 *This Recommendation defines the following services for the control of a single association*

a) A-ASSOCIATE;

b) A-RELEASE;

c) A-ABORT; and

d) A-P-ABORT.

7.2 The A-ASSOCIATE service causes the start of use of an association by those ASE procedures identified by the value of Application Context Name parameter.

Note – The use of an association by several ASEs is the subject of ongoing work.

7.3 The A-RELEASE service, if successful, causes the completion of the use of an association by those ASE procedures identified by the application context which is in effect without loss of information in transit. However, the success of the A-RELEASE service optionally may be negotiated.

7.4 The A-ABORT service causes the abnormal release of the association with the possible loss of information in transit.

7.5 The A-P-ABORT service indicates the abnormal release of the association as a result of action by the underlying presentation-service with the possible loss of information in transit.

7.6 For a particular association, the ACSE service operate in one of the following modes:

a) normal mode; or

b) X.410-1984 mode.

7.7 The normal mode of operation allows the ACSE service-user to take full advantage of the functionality provided by both ACSE and the presentation-service (X.216). In this mode the ACSE service-provider transfers its semantics using the normal mode of the presentation-service.

7.8 The X.410-1984 mode of operation allows the ACSE service-user to interwork with a peer using the protocol specified by the CCITT Recommendation X.410-1984. In this mode, the ACSE service-provider does not transfer any semantics of its own and uses the X.410-1984 mode of the presentation-service.

8 Relationship with other ASEs and lower layer services

8.1 *Other application-service-elements*

8.1.1 The ACSE is intended to be used with other ASEs in order to support a specific information processing task. Therefore, it is expected that the ACSE will be included in all application context specifications.

8.1.2 The collection of the ACSE and other ASE(s) included in an application context are required to use the facilities of the presentation-service in a coordinated manner.

8.2 *Presentation-service*

8.2.1 A one-to-one correspondence exists between an application-association and a presentation-connection.

8.2.2 The ACSE services require access to the P-CONNECT, P-RELEASE, P-U-ABORT and P-P-ABORT services. The ACSE services neither use nor constrain the use of any other presentation service.

8.2.3 The requestor and acceptor of the A-ASSOCIATE service determine the mode, the default presentation context, and the initial defined context set of the underlying presentation-connection using the following A-ASSOCIATE parameter:

- Mode;
- Presentation Requirements;
- Presentation Context Definition List;
- Presentation Context Definition Result List;
- Default Presentation Context Name; and
- Default Presentation Context Result.

8.2.4 If the requestor specifies the value "normal" for the Mode parameter, the last five parameters above determine the presentation context facility for the association according to the rules for the normal mode of the presentation-service (X.216). At the conclusion of the A-ASSOCIATE procedure, the requestor and acceptor must have obtained a presentation context which supports the abstract syntax specified in X.227 for the ACSE application-protocol-data-units.

Note – The ACSE service-provider is aware of the presentation context which contains its abstract syntax by a local mechanism.

8.2.5 If the requestor specifies the value "X.410-1984" for the Mode parameter, the ACSE service-provider does not transfer ACSE semantics and therefore does not require a presentation context for its abstract syntax. Any user information which the ACSE service-provider transfers for its service-user uses the unnamed default presentation context for the X.410-1984 mode of the presentation-service (X.216).

Note – Table 2/X.217 indicates the A-ASSOCIATE service parameters which are not used in the X.410-1984 mode. None of the presentation context related parameters are used.

8.3 *Session-service*

8.3.1 Using the Session Requirements parameter, the A-ASSOCIATE service requestor and acceptor determine the functional units for the underlying session-service (X.215).

8.3.2 The rules and parameter value length restrictions of the underlying session-service affect ACSE services. The ACSE service-user must be aware of these constraints.

Note – Some examples of these constraints are:

a) Version 1 of the session-protocol (X.225) imposes user data length restrictions which affect ACSE primitive parameters. Some special considerations apply to the A-ABORT services (see § 9.3).

b) The choice of session functional units for a particular association affects the rules for the use of ACSE services. For example, the selection of session tokens controls the possibilities of negotiated release and release collisions.

9 Service definition

The ACSE services are listed in Table 1/X.217.

TABLE 1/X.217

ACSE-services

Service	Type
A-ASSOCIATE	Confirmed
A-RELEASE	Confirmed
A-ABORT	Non-confirmed
A-P-ABORT	Provider-initiated

9.1 *A-ASSOCIATE Service*

The A-ASSOCIATE service is used to cause the beginning of the use of an association; it is a confirmed service.

9.1.1 *A-ASSOCIATE Parameters*

Table 2/X.217 lists the A-ASSOCIATE service parameters. In addition, groups of parameters are defined for reference by other ASEs as follows:

a) Calling AE Title is the composite of the Calling AP Title and the Calling AE Qualifier parameters;

b) Called AE Title is the composite of the Called AP Title and the Called AE Qualifier parameters;

c) Responding AE Title is the composite of the Responding AP Title and the Responding AE Qualifier parameters;

The two components of the AE title (AP title and AE qualifier) are defined in ISO 7498-3.

TABLE 2/X.217

A-ASSOCIATE parameters

Parameter name	Request	Indication	Response	Confirmation
Mode	U	M (=)		
Application context name [a)]	M	M (=)	M	M (=)
Calling AP title [a)]	U	C (=)		
Calling AE qualifier [a)]	U	C (=)		
Calling AP invocation-identifier [a)]	U	C (=)		
Calling AE invocation-identifier [a)]	U	C (=)		
Called AP title [a)]	U	C (=)		
Called AE qualifier [a)]	U	C (=)		
Called AP invocation-identifier [a)]	U	C (=)		
Called AE invocation-identifier [a)]	U	C (=)		
Responding AP title [a)]			U	C (=)
Responding AE qualifier [a)]			U	C (=)
Responding AP invocation-identifier [a)]			U	C (=)
Responding AE invocation-identifier [a)]			U	C (=)
User information	U	C (=)	U	C (=)
Result			M	M (=)
Result source				M
Diagnostic [a)]			U	C (=)
Calling presentation address	P	P		
Called presentation address	P	P		
Responding presentation address			P	P
Presentation context definition list [a)]	P	P		
Presentation context definition result list [a)]		P	P	P
Default presentation context name [a)]	P	P		
Default presentation context result [a)]			P	P
Quality of service	P	P	P	P
Presentation requirements [a)]	P	P	P	P
Session requirements	P	P	P	P
Initial synchronization point serial number	P	P	P	P
Initial assignment of tokens	P	P	P	P
Session-connection identifier [a)]	P	P	P	P

[a)] Not used in X.410-1984 mode.

9.1.1.1 *Mode*

This parameter specifies the mode in which the ACSE services will operate for this association. It takes one of the following symbolic values:

- normal; or
- X.410-1984.

If this parameter is not included on the request primitive, the default value of "normal" is used by the ACSE service provider. This parameter is always present on the indication primitive.

9.1.1.2 *Application Context Name*

This parameter identifies the application context proposed by the requestor. The acceptor returns either the same or a different name. The returned name specifies the application context to be used for this association.

Note – The offer of an alternate application context by the acceptor provides a possible mechanism for limited negotiation. However, the semantics and rules for this exchange are entirely user specific. If the requestor cannot operate in the acceptor's application context, it may issue an A-ABORT request primitive.

9.1.1.3 *Calling AP Title*

This parameter identifies the AP which contains the requestor of the A-ASSOCIATE service.

9.1.1.4 *Calling AE Qualifier*

This parameter identifies the particular AE of the AP which contains the requestor of the A-ASSOCIATE service.

9.1.1.5 *Calling AP Invocation-identifier*

This parameter identifies the AP invocation which contains the requestor of the A-ASSOCIATE service.

9.1.1.6 *Calling AE Invocation-identifier*

This parameter identifies the AE invocation which contains the requestor of the A-ASSOCIATE service.

9.1.1.7 *Called AP Title*

This parameter identifies the AP which contains the intended acceptor of the A-ASSOCIATE service.

9.1.1.8 *Called AE Qualifier*

This parameter identifies the particular AE of the AP which contains the intended acceptor of the A-ASSOCIATE service.

9.1.1.9 *Called AP Invocation-identifier*

This parameter identifies the AP invocation which contains the intended acceptor of the A-ASSOCIATE service.

9.1.1.10 *Called AE Invocation-identifier*

This parameter identifies the AE invocation which contains the intended acceptor of the A-ASSOCIATE service.

9.1.1.11 *Responding AP Title*

This parameter identifies the AP which contains the actual acceptor of the A-ASSOCIATE service.

9.1.1.12 *Responding AE Qualifier*

This parameter identifies the particular AE of the AP which contains the actual acceptor of the A-ASSOCIATE service.

9.1.1.13 *Responding AP Invocation-identifier*

This parameter identifies the AP invocation which contains the actual acceptor of the A-ASSOCIATE service.

9.1.1.14 *Responding AE Invocation-identifier*

This parameter identifies the AE invocation which contains the actual acceptor of the A-ASSOCIATE service.

9.1.1.15 *User Information*

Either the requestor or the acceptor may optionally include user information. Its meaning depends on the application context which accompanies the primitive.

Note – For example, this parameter may be used to carry the initialization information of other ASEs included in the application context specified by the value of the accompanying Application Context Name parameter.

9.1.1.16 *Result*

This parameter[3)] is provided by either the acceptor, by the ACSE service-provider, or by the presentation service-provider. It indicates the result of using the A-ASSOCIATE service. It takes one of the following symbolic values:

- accepted;
- rejected (permanent); or
- rejected (transient).

9.1.1.17 *Result Source*

The value of the parameter[3)] is supplied by the ACSE service-provider. It identifies the creating source of the Result parameter and the Diagnostic parameter, if present. It takes one of the following symbolic values:

- ACSE service-user;
- ACSE service-provider; or
- presentation service-provider.

If the Result parameter has the value "accepted", the value of this parameter is "ACSE service-user".

9.1.1.18 *Diagnostic*

This parameter[3)] is only used if the Result parameter has the value of "rejected (permanent)" or "rejected (transient)". Optionally, it can be used to provide diagnostic information about the result of the A-ASSOCIATE service.

If the Result Source parameter has the value "ACSE service-provider", it takes one of the following symbolic values:

- no reason given; or
- no common ACSE version.

If the Result Source parameter has the value "ACSE service-user", it takes one of the following symbolic values:

- no reason given;
- application context name not supported;
- calling AP title not recognized;
- calling AE qualifier not recognized;
- calling AP invocation-identifier not recognized;
- calling AE invocation-identifier not recognized;
- called AP title not recognized;
- called AE qualifier not recognized;
- called AP invocation-identifier not recognized; or
- called AE invocation-identifier not recognized.

[3)] It is recognized that, with respect to this parameter, work is still in progress to provide an integrated treatment across all layers of the OSI Reference Model. As a consequence, a change may be made to this Recommendation at a later time which reflects further developments and integration.

9.1.1.19 *Calling Presentation Address*

This parameter is as defined in Recommendation X.216.

9.1.1.20 *Called Presentation Address*

This parameter is as defined in Recommendation X.216.

9.1.1.21 *Responding Presentation Address*

This parameter is as defined in Recommendation X.216.

9.1.1.22 *Presentation Context Definition List*

This parameter is as defined in Recommendation X.216.

9.1.1.23 *Presentation Context Definition Result List*

This parameter is as defined in Recommendation X.216.

9.1.1.24 *Default Presentation Context Name*

This parameter corresponds to the Default Context Name parameter defined in Recommendation X.216.

9.1.1.25 *Default Presentation Context Result*

This parameter corresponds to the Default Context Result parameter defined in Recommendation X.216.

9.1.1.26 *Quality of Service*

This parameter is as defined in Recommendation X.216.

9.1.1.27 *Presentation Requirements*

This parameter is as defined in Recommendation X.216.

9.1.1.28 *Session Requirements*

This parameter is as defined in Recommendation X.216.

9.1.1.29 *Initial Synchronization Point Serial Number*

This parameter is as defined in Recommendation X.216.

9.1.1.30 *Initial Assignment of Tokens*

This parameter is as defined in Recommendation X.216.

9.1.1.31 *Session Connection Identifier*

This parameter is as defined in Recommendation X.216.

9.1.2 *A-ASSOCIATE service procedure*

9.1.2.1 The A-ASSOCIATE service procedure has a one-to-one correspondence with the P-CONNECT service defined in Recommendation X.216. When the A-ASSOCIATE service is used, the association is created simultaneously with the creation of the underlying presentation-connection.

9.1.2.2 An ACSE service-user which desires to establish an association issues an A-ASSOCIATE request primitive. The called AE is identified by parameters of the request primitive. The requestor cannot issue any primitives except an A-ABORT request primitive until it receives an A-ASSOICATE confirm primitive.

9.1.2.3 The ACSE service-provider issues an A-ASSOCIATE indication primitive to the acceptor.

9.1.2.4 The acceptor accepts or rejects the association by sending an A-ASSOCIATE response primitive with an appropriate Result parameter. ACSE service-provider issues an A-ASSOCIATE confirm primitive having the same Result parameter. The Result Source parameter is assigned the symbolic value of "ACSE service-user".

9.1.2.5 If the acceptor accepts the association, the association is available for use. Requestors in both AEs may now use any service provided by the ASEs included in the application context which in effect (with the exception of A-ASSOCIATE).

9.1.2.6 If the acceptor rejects the association, the association is not established.

9.1.2.7 The ACSE service-provider may not be capable of supporting the requested association. In this situation, it returns an A-ASSOCIATE confirm primitive to the requestor with an appropriate RESULT parameter. The Result Source parameter is appropriately assigned either the symbolic value of "ACSE service-provider" or "presentation service-provider". The indication primitive is not issued. The association is not established.

9.1.2.8 A requestor in either AE may disrupt the A-ASSOCIATE service procedure by issuing an A-ABORT request primitive. The acceptor receives an A-ABORT indication primitive. The association is not established.

9.2 *A-Release service*

The A-RELEASE service is used by a requestor in either AE to cause the completion of the use of an association; it is a confirmed service. If the session Negotiated Release functional unit was selected for the association, the acceptor may respond negatively (see § 8.3.2). This causes the unsuccessful completion of the A-RELEASE service and the continuation of the association without loss of information in transit.

9.2.1 *A-RELEASE parameters*

Table 3/X.217 lists the A-RELEASE parameters.

TABLE 3/X.217

A-RELEASE parameters

Parameter name	Request	Indication	Response	Confirmation
Reason [a]	U	C(=)	U	C(=)
User information [a]	U	C(=)	U	C(=)
Result			M	M(=)

[a] Not used in X.410-1984 mode.

9.2.1.1 *Reason*

When used on the request primitive, this parameter identifies the general level of urgency of the request. It takes one of the following symbolic values:

- normal;
- urgent; or
- user defined.

Note – For example, if the session Negotiated Release functional unit was selected for the association, the value "urgent" may be used on the request primitive when the requestor desires to urgently release the association.

When used on the response primitive, this parameter identifies information about why the acceptor accepted or rejected the release request. It takes one of the following symbolic values:

- normal;
- not finished; or
- user defined.

Note – For example, if the session Negotiated Release functional unit was not selected for the association, the value "not finished" may be used on the response primitive when the acceptor is forced to release the association but wishes to give a warning that it has additional information to send or receive.

9.2.1.2 *User information*

Either the requestor or acceptor may optionally include user information on the request or response primitive. Its meaning depends on the application context which is in effect.

9.2.1.3 *Result*

This parameter is used by the acceptor to indicate if the request to release the association normally is acceptable. It takes one of the following symbolic values:

- affirmative; or
- negative.

9.2.2 *A-RELEASE service procedure*

9.2.2.1 The A-RELEASE service procedure has a one-to-one correspondence with the P-RELEASE service defined in Recommendation X.216. When the A-RELEASE service is used, the association is released simultaneously with the release of the underlying presentation-connection.

9.2.2.2 An ACSE service-user which desires to release the association issues an A-RELEASE request primitive. This requestor cannot issue any further primitives other than an A-ABORT request primitive until it receives an A-RELEASE confirm primitive.

9.2.2.3 In order to issue an A-RELEASE request primitive, the requestor is required to meet all the requirements for issuing a P-RELEASE request (see § 8.2).

9.2.2.4 The ACSE service-provider issues an A-RELEASE indication primitive to the acceptor. The acceptor then cannot issue any ACSE primitives other than an A-RELEASE response primitive or an A-ABORT request primitive.

9.2.2.5 The acceptor replies to the A-RELEASE indication primitive by issuing an A-RELEASE response primitive with a Result parameter which has a value of "affirmative" or "negative". The acceptor may give a negative response only if the session Negotiated Release functional unit was selected for the association (see § 8.3).

9.2.2.6 If the acceptor gives a negative reponse, it may once again use any service provided by the ASEs included in the application context which in effect (with the exception of the A-ASSOCIATE service). If it gave a positive response, it cannot issue any further primitives for the association.

9.2.2.7 The ACSE service-provider issues an A-RELEASE confirm primitive with an "affirmative" or "negative" value for the Result parameter. If the value is "negative", the requestor may once again use any of the services provided by the ASEs of the application context which is in effect (with the exception of A-ASSOCIATE).

9.2.2.8 If the value of the Result parameter is "affirmative", the association and the underlying presentation-connection have been released.

9.2.2.9 A requestor in either AE may disrupt the A-RELEASE service procedure by issuing an A-ABORT request. The acceptor receives an A-ABORT indication. The association is released with the possible loss of information in transit.

9.2.2.10 An A-RELEASE service procedure collision results when requestors in both AEs simultaneously issue an A-RELEASE service primitive. This can occur only when no session tokens are available on the association (see § 8.3). In this situation, both ACSE service-users receive an unexpected A-RELEASE indication primitive. The following sequence then occurs to complete the normal release of the association.

a) The association-initiator issues an A-RELEASE response primitive.

b) The association-responder waits for an A-RELEASE confirm primitive from its peer. When it receives one, it then issues an A-RELEASE response primitive.

c) The association-initiator receives an A-RELEASE confirm primitive.

9.2.2.11 The association is released when both ACSE service-users have received an A-RELEASE confirm primitive.

9.3 *A-ABORT service*

The A-ABORT service is used by a requestor in either AE to cause the abnormal release of the association. It is a non-confirmed service. However, because of the possibility of an A-ABORT service procedure collision (see § 10.3.5), the delivery of the indication primitive is not guaranteed. However, both AEs are aware that the association has been released.

9.3.1 *A-ABORT parameters*

Table 4/X.217 lists the A-ABORT parameters.

TABLE 4/X.217

A-ABORT parameters

Parameter name	Request	Indication
Abort source [a]		M
User information	U	C(=)

[a] Not used in X.410-1984 mode.

9.3.1.1 *Abort Source*

This parameter, whose value is supplied by the ACSE service-provider, indicates the initiating source of this abort. It takes one of the following symbolic values:

- ACSE service-user; or
- ACSE service-provider.

9.3.1.2 *User Information*

The requestor may optionally include user information on the request primitive. Its meaning depends on the application context which is in effect.

Note – When ACSE is supported with version 1 of the session-protocol (X.225), this parameter is subject to length restrictions mentioned in § 8.3. For use with version 1, the A-ABORT service procedure does not transfer any of its own semantics, thus allowing the maximum possible length for presentation data value(s) of the User Information parameter. In this situation, the Abort Source parameter of the A-ABORT indication primitive always indicates "ACSE service-user".

9.3.2 *A-ABORT service procedure*

9.3.2.1 The A-ABORT service procedure has a one-to-one correspondence with the P-U-ABORT service defined in Recommendation X.216. When the A-ABORT service is used, the association is abnormally released simultaneously with the abnormal release of the underlying presentation-connection.

9.3.2.2 An ACSE service-user which desires to abnormally release the association issues the A-ABORT request primitive. This requestor cannot issue any further primitives for the association.

9.3.2.3 The ACSE service-provider issues an A-ABORT indication primitive to the acceptor. The ACSE service-provider assigns the value of "ACSE service-user" for the Abort Source parameter. The association and the underlying presentation-connection have been released.

9.3.2.4 The ACSE service-provider may itself cause the abnormal release of the association because of internal errors detected by it. In this case, the ACSE service-provider issues A-ABORT indication primitives to acceptors in both AEs. The ACSE service-provider assigns the value of "ACSE service-provider" to the Abort Source parameter. The User Information parameter is not used.

9.4 *A-P-ABORT service*

The A-P-ABORT service is used by the ACSE service-provider to signal the abnormal release of the association due to problems in services below the Application Layer. This occurrence indicates the possible loss of information in transit. A-P-ABORT is a provider-initiated service.

9.4.1 *A-P-ABORT parameter*

Table 5/X.217 lists the A-P-ABORT parameter.

TABLE 5/X.217

A-P-ABORT parameter

Parameter name	Indication
Provider reason	P

Provider Reason: This parameter is as defined in Recommendation X.216.

9.4.2 *A-P-ABORT service procedure*

When the ACSE service-provider detects an error reported by the underlying presentation-service, it issues A-P-ABORT indication primitives to acceptors in both AEs. The association is abnormally released. Requestors in both AEs cannot issue any further primitives for the association.

10 Sequencing information

This clause defines the interaction among the ACSE service procedures for a particular association.

10.1 *A-ASSOCIATE*

10.1.1 *Type of service*

A-ASSOCIATE is a confirmed service.

10.1.2 *Usage restrictions*

The A-ASSOCIATE service is not used on an established association.

10.1.3 *Disrupted service procedures*

The A-ASSOCIATE service procedure does not disrupt any other service procedure.

10.1.4 *Disrupting service procedures*

The A-ASSOCIATE service procedure is disrupted by the A-ABORT service procedures.

10.1.5 *Collisions*

An A-ASSOCIATE service procedure collision results when requestors in both AEs simultaneously issue an A-ASSOCIATE request primitive for each other. Both ACSE service-users are issued A-ASSOCIATION indication primitives which represent distinct associations. Both can choose to accept or reject the indicated association by issuing an A-ASSOCIATE response primitive with the appropriate value for its Result parameter. This will result in the establishment of none, one or two associations.

Note – If an AE has several concurrent associations, a local mechanism is needed to distinguish between them.

10.2 *A-RELEASE*

10.2.1 *Type of service*

A-RELEASE is a confirmed service.

10.2.2 *Usage restrictions*

The A-RELEASE service is only used on an established association.

10.2.3 *Disrupted service procedures*

The A-RELEASE service procedure does not disrupt any other service procedure, except that an A-ABORT indication is suppressed following issuance of an A-RELEASE response, and that an A-P-ABORT indication is suppressed following issuance of an A-RELEASE response or confirm.

10.2.4 *Disrupting service procedures*

The A-RELEASE service procedure is disrupted by the A-ABORT or A-P-ABORT service procedures.

10.2.5 *Collisions*

An A-RELEASE service procedure collision results when requestors in both AEs simultaneously issue an A-RELEASE request primitive. The processing of A-RELEASE service procedure collisions is described in § 9.2.2.

10.2.6 *Further sequencing information*

The use of the A-RELEASE service is subject to the constraints on the S-RELEASE service defined in Recommendation X.215 (see § 8.3).

10.3 *A-ABORT*

10.3.1 *Type of service*

A-ABORT is a non-confirmed service.

10.3.2 *Usage restrictions*

The A-ABORT service has effect only when used on an association in the process of establishment, on an established association, or on an association in the process of being released.

10.3.3 *Disrupted service procedures*

The A-ABORT service procedure disrupts the A-ASSOCIATE, A-RELEASE and A-P-ABORT service procedures.

10.3.4 *Disrupting service procedures*

The A-ABORT service procedure is disrupted by the A-P-ABORT service procedure and by the issuance of an A-RELEASE response.

10.3.5 *Collisions*

An A-ABORT service procedure collision results when requestors in both AEs simultaneously issue the A-ABORT request primitive. The collision processing is governed by the P-U-ABORT service defined in Recommendation X.216. In this situation, neither A-ABORT indication primitive is issued.

10.3.6 *Further sequencing information*

Any use of the A-ABORT service results in the abnormal release of the association, or the abnormal termination of the A-ASSOCIATE service procedure or the A-RELEASE service procedure with possible loss of information.

10.4 *A-P-ABORT*

10.4.1 *Type of service*

A-P-ABORT is a provider-initiated service.

10.4.2 *Usage restrictions*

No restrictions are placed on the occurrence of this service.

10.4.3 *Disrupted service procedures*

The A-P-ABORT service procedure disrupts all other service procedures.

10.4.4 *Disrupting service procedures*

The A-P-ABORT service procedure is disrupted by the A-ABORT service procedure and by the issuance of an A-RELEASE response or confirm.

ANNEX A

(to Recommendation X.217)

Usage of ACSE services to achieve compatibility with CCITT Recommendation X.410-1984, and the basic facilities of the 1988 Message Handling series of CCITT Recommendations

A.1 *Compatibility requirements*

Recommendation X.410, which was adopted by CCITT in 1984, has been used in a number of Recommendation X.400 Message Handling products available or under development.

It is essential that the systems following Recommendation X.410-1984 be able to interwork with OSI systems. This principle has been guiding the development of the OSI ACSE and Presentation service and protocol, which has been conducted in very close cooperation between CCITT and ISO.

This Annex shows how the ACSE service is to be used to achieve compatibility with Recommendation X.410-1984, and to support the basic facilities of the 1988 Message Handling series of CCITT Recommendations.

Reference is also made to a companion Annex attached to the OSI Presentation service (Recommendation X.216) which shows how OSI Presentation service is to be used in order to achieve compatibility with Recommendation X.410-1984.

A.2 *Principles for ensuring compatibility*

The ACSE X.410-1984 mode of operation can be activated resulting in full encoding alignment with X.410-1984 at the session user data level. Its effect is to suppress the generation of explicit ACSE APDUs while maintaining an active ACPM state machine (see Recommendation X.227, Annex B).

The layered structure of both protocols, which conforms with the OSI Reference Model, makes it possible to distinguish the Presentation Layer functions and parameters from those of the Application Layer. Based on this layering, the following principles are used to ensure the required compatibility:

a) The functions and the corresponding protocol elements of X.410-1984 which belong to the Presentation Layer are integrated into the OSI Presentation protocol, which remains consistent and satisfies the requirements of the whole set of OSI applications.

b) The additional functions and protocol elements of X.410-1984 are placed in the Application Layer. They are generated by the Reliable Transfer Service Element (RTSE, see Recommendations X.218 and X.228 also Recommendation X.410-1984). They are passed transparently by the ACSE service-provider during association establishment and release by making direct use of the Presentation services.

A.3 *Usage of Association Control services to ensure compatibility with Recommendation X.410-1984*

The following Association Control services are used:

A-ASSOCIATE

A-RELEASE

A-ABORT

A-P-ABORT

The use of these services is explained in §§ A.3.1-A.3.5.

Note – RTORQ, RTOAC, RTORJ and RTAB are names given to APDUs generated by the RTSE protocol machine.

A.3.1 *A-ASSOCIATE*

An association is established and the X.410-1984 mode of ACSE operation is activated with the following combination of A-ASSOCIATE service parameters:

a) Mode parameter must be set to "X.410-1984" on request primitive;

b) Presentation Requirements parameter must specify the kernel.

c) Session Requirements parameter must be set according to X.410-1984; and

d) User Information:

1) On the request and indication primitives, this parameter must contain an RTORQ APDU.

2) On the response and confirmation primitives, it must contain a RTOAC APDU if the association has been accepted, or a RTORJ APDU if the association has been rejected.

3) If the ACSE service-provider has rejected the request, then this parameter is not used.

All other A-ASSOCIATE parameters must be absent or as defined by the Presentation Service and its Annex concerning its use for X.410-1984 compatibility (Recommendation X.216).

Following the occurrence of an A-ABORT or A-P-ABORT service event, the initiating RTSE may use the A-ASSOCIATE service an implementation dependent number of times to attempt recovery. This use of the service has all parameters absent except for Presentation User Data which must contain the recovery data from the RT-OPEN service.

A.3.2 *A-RELEASE*

The association is released by the use of this service with only the Result parameter present. The rules governing the use of A-RELEASE are as in the main body of this document and are identical to those for P-RELEASE.

A.3.3 *A-ABORT*

Either ACSE service-user may signal to its peer that it has a problem and attempt to force the release of an association by using A-ABORT service with all parameters absent except for the Presentation User data parameter, which must contain an RTAB APDU. The association is released, and the initiating RTSE may use the A-ASSOCIATE service to attempt to obtain a new association over which it can execute its recovery procedures.

A.3.4 *A-P-ABORT*

Either ACSE service-provider may signal that it has a problem (internally or with the services which underlie it) to its peer and force the release of an association by using the A-P-ABORT service as defined in Recommendation X.216. The association is released, and the initiating RTSE may use the A-ASSOCIATE service to attempt to obtain a new association over which it can execute its recovery procedures.

A.3.5 *State Table*

The state table which governs the operation of the ACSE in X.410-1984 mode is given in Annex B of Recommendation X.227.

APPENDIX I

(to Recommendation X.217)

Differences between Recommendation X.217 and ISO International Standard 8649

I.1 Recommendation X.217 and ISO 8649 are technically aligned, with the following minor exceptions:

I.2 In § 10 on Sequencing Information this Recommendation states that the A-ABORT and A-P-ABORT services are mutually disruptive, when they collide (see §§ 10.3.3 and 10.4.4). ISO 8649 states that A-P-ABORT cannot be disrupted (see §§ 10.3.3 and 10.4.4).

I.3 In § 10 on Sequencing Information this Recommendation states that an A-ABORT indication is suppressed following issuance of an A-RELEASE response, and that an A-P-ABORT indication is suppressed following issuance of an A-RELEASE response or confirm (see §§ 10.2.3, 10.3.4 and 10.4.4). ISO 8649 states that the A-RELEASE service procedure does not disrupt any other service procedure (see §§ 10.2.3, 10.3.4 and 10.4.4).

I.4 This Recommendation contains an Annex A, which has not been included in ISO 8649. Annex A shows how the OSI Association Control service is to be used in order to achieve compatibility with Recommendation X.410-1984 and to support the basic facilities of the 1988 message handling services of CCITT Recommendations (the X.400 series).

I.5 There is no equivalent of this Appendix I in ISO 8649.

Recommendation X.218

RELIABLE TRANSFER: MODEL AND SERVICE DEFINITION[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.228 defines the Reliable Transfer protocol;

(f) that there is a need for common Reliable Transport support for various applications,

unanimously declares

that this Recommendation defines the Reliable Transfer service of Open Systems Interconnection for CCITT Applications as given in the Scope and Field of Application.

CONTENTS

[1)] Recommendation X.218 and ISO 9066-1 [Information processing systems – Text Communication – Reliable Transfer Part 1: Model and Service Definition] were developed in close collaboration and are technically aligned.

0 Introduction

This Recommendation defines the services provided by 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 defining 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 9066-1.

1 Scope and field of application

This Recommendation defines the services provided by the Reliable Transfer Service Element (RTSE). The RTSE services are provided by the use of the RTSE protocol (Recommendation X.228) 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).

No requirement is made for conformance to this Recommendation.

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.227	Association Control Protocol Specification for CCITT Applications (see also ISO 8650).
Recommendation X.228	Reliable Transfer: Protocol Specification (see also ISO 9066-2).

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 Layer;
h) presentation-service;
i) presentation-connection;
j) session-service;
k) session-connection;
l) transfer syntax;
m) two-way-alternate interaction; and
n) user-element.

3.2 *Service convention 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) service-primitive; 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) default context;
d) presentation context;
e) transfer syntax name.

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;
d) X.410-1984 mode.

3.5 *Reliable transfer definitions*

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

3.5.1 **association-initiating-application-entity; association-initiator**

The application-entity that initiates the application-association.

3.5.2 **association-responding-application-entity; association-responder**

The application-entity that responds to the initiation of an application-association by another AE.

3.5.3 **sending-application-entity; sender**

The application-entity that sends, or may send (i.e., possesses the TURN) the APDU to the receiving application-entity.

3.5.4 **receiving-application-entity; receiver**

The application-entity that receives, or may receive (i.e., does not possess the Turn) the APDU from the sending application-entity.

3.5.5 **requestor**

The part of an application-entity that issues a request primitive, or receives a confirm primitive for a particular RTSE service.

3.5.6 **acceptor**

The part of an application-entity that receives the indication primitive, or issues a response primitive for a particular RTSE service.

3.5.7 **Reliable Transfer Service Element**

The application-service-element defined in this Recommendation.

3.5.8 **Reliable Transfer**

An application-independent mechanism to provide for the transfer of application-protocol-data-units between open systems, and to recover from communication and end-system failure minimizing the amount of retransmission.

3.5.9 **RTSE-user**

The user of the Reliable Transfer Service Element. The user may be the user element, or another application service element, of the application entity.

3.5.10 **RTSE-provider**

The provider of the Reliable Transfer Service Element.

3.5.11 **ACSE-provider**

The provider of the Association Control Service Element.

3.5.12 **monologue interaction**

A mode of interaction where only one application-entity may be the sender.

3.5.13 **syntax-matching services**

Local services provided by the presentation-service provider enabling the transformation from the local representation of an application-protocol-data-unit value into a representation specified by a negotiated transfer syntax and vice versa.

3.5.14 **X.410-1984 mode**

A restricted mode of operation of the Reliable Transfer Service Element to allow interworking with application-entities based on CCITT Recommendation X.410-1984.

3.5.15 **normal mode**

A mode of operation of the Reliable Transfer Service Element providing full services.

4 Abbreviations

AE	application entity
ACSE	Association Control Service Element
APDU	application-protocol-data-unit
ASE	application-service-element
OSI	Open Systems Interconnection
RT (or RTS)	Reliable Transfer
RTSE	Reliable Transfer Service Element

5 Conventions

This Recommendation defines services for the RTSE following the descriptive conventions defined in Recommendation X.210. In § 9, the definition of each RTSE service includes a table that lists the parameters of its primitives. For a given primitive, the presence of each parameter is described by one of the following values.

Blank not applicable
M mandatory
U user option
C conditional

T presence is an RTSE-provider option

A presence subject to conditions defined in Recommendation X.217

P presence subject to conditions defined in Recommendation X.216

In addition, the notation (=) indicates that a parameter value is semantically equal to the value to its left in the table.

6 Reliable Transfer model

In the OSI environment, communication between application-processes is represented in terms of communication between a pair of application-entities (AEs) using the presentation-service. Communication between some application-entities requires the Reliable Transfer of application-protocol-data-units (APDUs).

APDUs sent by one AE (the sender) are received by the other AE (the receiver). Reliable Transfer ensures that each APDU is completely transferred between AEs exactly once, or that the sending AE is warned of an exception. Reliable Transfer recovers from communication and end-system failure and minimizes the amount of retransmission needed for recovery. The APDUs transferred are transparent to the Reliable Transfer.

Reliable Transfer is carried out within the context of an application-association. An application-association defines the relationship between a pair of AEs, and is formed by the exchange of application-protocol-control-information through the use of presentation-services. The AE that initiates an application-association is called the association-initiating AE, or the association-initiator, while the AE that responds to the initiation of an application-association by another AE is called the association-responding AE, or the association-responder. Only the association-initiator may release an established application-association.

The functionality of an AE is factored into one user-element and a set of association-service-elements (ASEs). Each ASE may itself be factored into a set of (more primitive) ASEs. The interaction between AEs is described in terms of their use of ASEs.

The specific combination of a user-element and the set of ASEs which comprise an AE are defined by the association context.

Figure 1/X.218 illustrates an example of an application context involving the Reliable Transfer Service Element (RTSE).

The ASEs available to the user-element require communication over an application-association. The control of that application-association (establishment, release, abort) and the Reliable Transfer of APDUs over the application-association is performed by the Reliable Transfer Service Element (RTSE) defined in this Recommendation. The RTSE uses the Association Control Service Element (ACSE) defined in Recommendation X.217 for control of that application-association (establishment, release, abort).

Note that the application context depicted in Figure 1/X.218 is minimal for an application context involving RTSE. Another example, taken from message handling (Recommendation X.400), of an application context involving RTSE, could be that of a message transfer agent (MTA), and would include the message transfer service element (MTSE) in addition to the ACSE and the RTSE. Note also that, in general, it is the responsibility of a Recommendation defining a set of ASEs that make use of the RTSE (and the ACSE), to define what use is made of the RTSE and any restrictions that may apply.

7 Overview of service

This Recommendation defines the following services for Reliable Transfer:

a) RT-OPEN

b) RT-CLOSE

c) RT-TRANSFER

d) RT-TURN-PLEASE

e) RT-TURN-GIVE

f) RT-P-ABORT

g) RT-U-ABORT

The RT-OPEN service enables an RTSE-user to request the establishment of an application-association with another AE.

The RT-CLOSE service enables the association-initiating RTSE-user to request the release of an established application-association. It may do so only if it possesses the Turn.

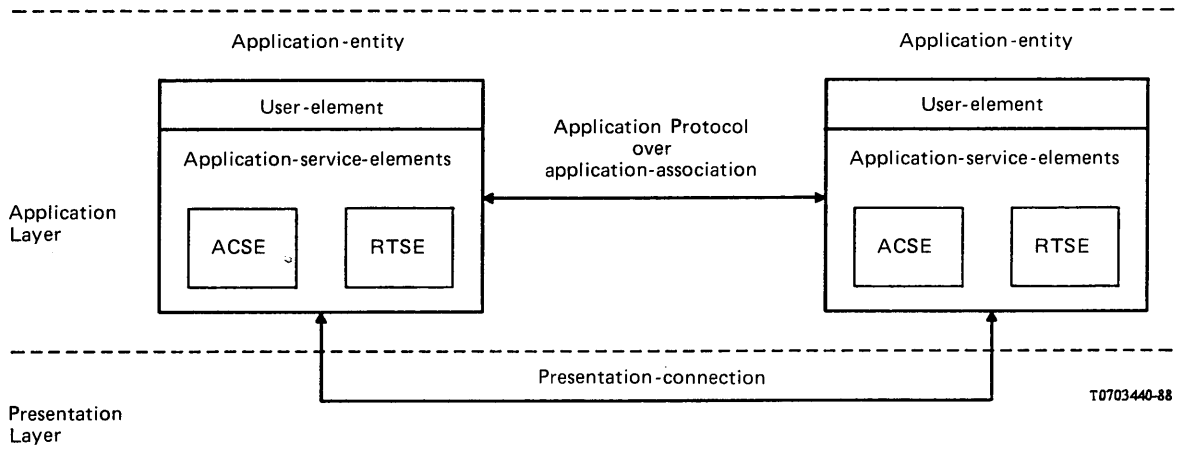


FIGURE 1/X.218

Model of an application context involving reliable transfer

The RT-TRANSFER service enables an RTSE-user that possesses the Turn, to request the Reliable Transfer of an APDU over an application-association. It may do so on an established application-association and where there is no outstanding RT-TRANSFER confirm positive.

The RT-TURN-PLEASE service enables an RTSE-user to request the Turn. It may do so only if it does not already possess the Turn. The Turn is requested by either RTSE-user to allow the RTSE-user to transfer APDUs. The Turn is requested by the association-initiating RTSE-user to allow it to release the application-association. The request conveys the priority of the action to be taken so that the other RTSE-user can decide when to actually relinquish the Turn.

The RT-TURN-GIVE service enables an RTSE-user to relinquish the Turn to its peer. It may do so only if it possesses the Turn.

The RT-P-ABORT service provides an indication to the RTSE-user that the application-association cannot be maintained (e.g., because recovery not possible, etc.). If it is the sender, the RTSE-provider first issues a negative RT-TRANSFER confirm for the APDU not yet transferred. If it is the receiver, the RTSE-provider deletes the partially received APDU prior to issuing the RT-P-ABORT indication.

the RT-U-ABORT service enables an RTSE-user to abort the application-association.

The Reliable Transfer is provided in two modes of operation:

a) X.410-1980 mode: is provided solely to allow interworking with older implementations based on CCITT Recommendation X.410-1984. This mode implies some restriction in the use of RTSE services;

b) normal mode: is provided to allow full use of RTSE services.

8 Relationship with other ASEs and Lower Layer services

8.1 *Other application service elements*

The RTSE is intended to be used with other ASEs in order to support specific information processing tasks that require the Reliable Transfer of application-protocol-data-units. Therefore, it is expected that the RTSE will be included in a number of application context specifications.

The collection of the RTSE and other ASEs (in particular ACSE) included in an application context are required to use the facilities of the presentation-service in a co-ordinated manner among themselves.

The RTSE requires the control of an application-association by the ACSE. For application contexts that involve RTSE, the RTSE-provider is the user of the A-P-ABORT service; the A-P-ABORT service is not used directly by the user-element nor by any other ASE. In the event of the RTSE-provider receiving an A-P-ABORT indication from the ACSE-provider, the RTSE-provider will attempt to recover the presentation-connection by issuing an A-ASSOCIATE request. If the presentation-connection cannot be recovered, the RTSE-provider will issue an RT-P-ABORT indication to the RTSE-user. The A-ABORT service provided by the ACSE is used by the RTSE-provider.

An RTSE-user protocol specification defines the type of User-data parameter values of the RTSE services forming one or more abstract syntaxes and provides an unique abstract syntax name of type object identifier for each abstract syntax.

The User-data parameter values (if any) for the RT-OPEN and RT-U-ABORT services shall share one single named abstract syntax with the RTSE APDUs defined in Recommendation X.228. The Types for User-data parameter values (if any) of the RT-OPEN request/confirm, RT-OPEN response/confirm positive, RT-OPEN response/confirm negative and RT-U-ABORT request/indication primitives shall be any single ASN.1 type each. If no types for User-data parameter values are defined, the abstract syntax name rTSE-abstract-syntax defined in Recommendation X.228 identifies an abstract syntax formed by the RTSE APDUs.

The types of User-data parameter values for the RT-CLOSE services (if any) and the RT-TRANSFER service may form one or more named abstract syntaxes. Within a single named abstract syntax the type shall be a single ASN.1 type usually (but not necessarily) a choice type. This types may share a single abstract syntax with the RTSE APDUs, if and only if they use tags distinct from context-specific tags with the numbers [16], [17], [18] and [22] and distinct from the ASN.1 integer type and octetstring type. These conditions are ensured, if the RTSE-user protocol uses the RO-notation of Recommendation X.219.

In X.410-1984 mode there exists only a single abstract syntax, however this abstract syntax is not identified by an abstract syntax name but by the value of the Application-protocol parameter value of the RT-OPEN service.

8.2 *ACSE services*

The RTSE services require access to the A-ASSOCIATE, A-RELEASE, A-ABORT, and A-P-ABORT services. The inclusion of the RTSE in an application context precludes the use of any of the above ACSE services by any other ASE or the user-element.

The X.410-1984 mode of RTSE implies the X.410-1984 mode of ACSE.

8.3 *Presentation-service*

The RTSE services require 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 recognizes that the ACSE services require access to the P-CONNECT, P-RELEASE, P-U-ABORT and P-P-ABORT services. The inclusion of the RTSE in an application context precludes the use of any of the above, or of any other, presentation-services by any other ASE or the user-element.

The RT protocol machine makes use of syntax-matching-services in the local system environment for its operation. These services are used to transform the representation of APDUs transferred between ASEs which use the RTSE. The syntax-matching-services provide for the transformation from a local representation of an APDU into a representation specified by a transfer syntax determined by the Presentation Service and vice versa. The method used to access this transfer syntax information is a local matter outside the scope of this Recommendation.

The X.410-1984 mode of RTSE implies the X.410-1984 mode of the Presentation-service.

A named abstract syntax associated with a compatible transfer syntax (negotiated by the Presentation Layer) constitutes a presentation context.

The object identifier value {joint-iso-ccitt asn1(1)basic-encoding(1)} specified in Recommendation X.209 may be used as a transfer syntax name. In this case the RTSE-user protocol specification need not name and specify a transfer syntax.

In X.410-1984 mode the default presentation context is constituted by the single abstract syntax identified by the Application-protocol parameter value of the RT-OPEN service associated with basic ASN.1 encoding rules of Recommendation X.209.

9 Service definition

The RTSE services are listed in Table 1/X.218.

TABLE 1/X.218

RTSE services

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

Identification of the named abstract syntax in use is assumed for all RTSE services; however, this is a local matter and outside the scope of this Recommendation.

9.1 *RT-OPEN service*

The RT-OPEN service is used by the association-initiator to request the establishment of an application-association for the ASE procedures identified by the Application Context Name parameter (in normal mode), or by the Application-protocol parameter (in X.410-1984 mode). This service is a confirmed service.

The related service structure consists of four service primitives, as illustrated in Figure 2/X.218.

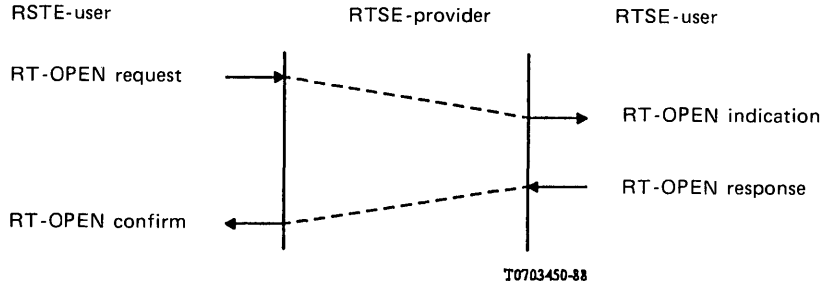


FIGURE 2/X.218

RT-OPEN Service-primitives

9.1.1 *Parameters of RT-OPEN*

Table 2/X.218 lists the RT-OPEN service parameters.

TABLE 2/X.218

RT-OPEN parameters

Parameter name		Request	Indication	Response	Confirmation
Dialogue-mode		M	M (=)		
Initial-turn		M	M (=)		
Application-protocol	4)	U	C (=)		
User-data	2)	U	C (=)	U	C (=)
Mode		A	A		
Application context name	3)	A	A	A	A
Calling AP title	3)	A	A		
Calling AP invocation-identifier	3)	A	A		
Calling AE qualifier	3)	A	A		
Calling AE invocation-identifier	3)	A	A		
Calling AP title	3)	A	A		
Called AP invocation-identifier	3)	A	A		
Calling AE qualifier	3)	A	A		
Called AE invocation-identifier	3)	A	A		
Responding AP title	3)			A	A
Responding AP invocation-identifier	3)			A	A
Responding AE qualifier	3)			A	A
Responding AE invocation-identifier	3)			A	A
Result				A	A
Result source					A
Diagnostic				A	A
Calling presentation address		P	P		
Called presentation address		P	P		
Responding presentation address				P	P
Presentation context definition list	3)	P	P		
Presentation context definition result list	3)		P	P	P
Default presentation context name	3)	P	P		
Default presentation context result	3)		P	P	P

Note 1 – If the parameter has the value "X.410-1984 mode" the X.410-1984 mode applies.

Note 2 – Restricted use of parameters in X.410-1984 mode (see following clauses).

Note 3 – Parameter absent in X.410-1984 mode.

Note 4 – Parameter only present in X.410-1984 mode.

9.1.1.1 *Dialogue-mode*

The type of use of the application-association:

- monologue, or
- two-way-alternate interaction.

9.1.1.2 *Initial turn*

The RTSE-user that is to have the turn initially:

- association-initiator, or
- association-responder.

9.1.1.3 *Application-protocol*

Designates the application protocol that will govern communication over the application-association.

This parameter is only present in X.410-1984 mode. In normal mode the parameter Application Context Name is used.

9.1.1.4 *User-data*

User data associated with establishing the application-association.

If the X.410-1984 mode is selected, and the Result parameter of the RT-OPEN response primitive has the value "rejected (permanent)", this parameter in the RT-OPEN response primitive is restricted to the values:

- authentication-failure, and
- unacceptable-dialogue-mode.

If the X.410-1984 mode is selected, and the Result parameter of the RT-OPEN response primitive has the value "rejected (transient)", this parameter in the RT-OPEN response primitive is absent.

In the normal mode the use of this parameter is not restricted.

9.1.1.5 *Mode*

This parameter specifies the mode in which the RTSE services will operate for this association. It takes one of the following symbolic values:

- normal mode, or
- X.410-1984 mode.

9.1.1.6 *Other parameters*

Parameters marked with an "A" in Table 2/X.218 are defined in Recommendation X.217.

Parameters marked with a "P" in Table 2/X.218 are defined in Recommendation X.216.

9.2 *RT-Close service*

The RT-CLOSE service is used by the association-initiator to request the release of an application-association. It may do so only if it possesses the Turn and there is no outstanding RT-TRANSFER confirm primitive. This service is a confirmed service.

The release of the application-association is without loss of information in transit. This service can not be rejected by the association-responding RTSE-user.

The related service structure consists of four service-primitives, as illustrated in Figure 3/X.218.

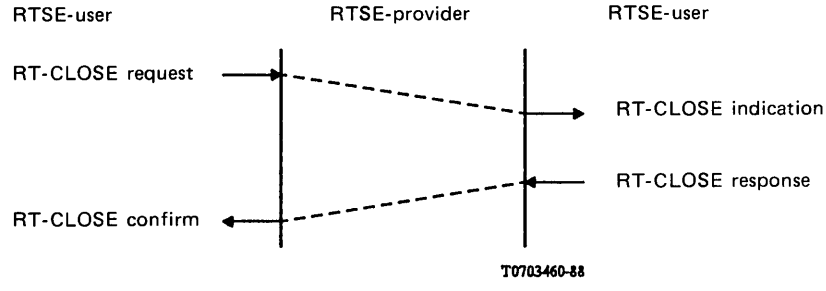


FIGURE 3/X.218

RT-CLOSE Service-primitives

9.2.1 *Parameters of RT-CLOSE*

Table 3/X.218 lists the RT-CLOSE service parameters. These parameters are only present in the normal mode and are defined in Recommendation X.217. In the X.410-1984 mode the RT-CLOSE service has no parameters.

TABLE 3/X.218

RT-CLOSE parameters

Parameter name	Request	Indication	Response	Confirmation
Reason	A	A	A	A
User-data	A	A	A	A

9.3 *RT-TRANSFER service*

The RT-TRANSFER service enables an RTSE-user that possesses the Turn, to request the Reliable Transfer of an APDU over an application-association. It may do so only on an established application-association and when there is no outstanding RT-TRANSFER confirm primitive. This service is a confirmed service.

The related service structure consists of three service-primitives, as illustrated in Figure 4/X.218.

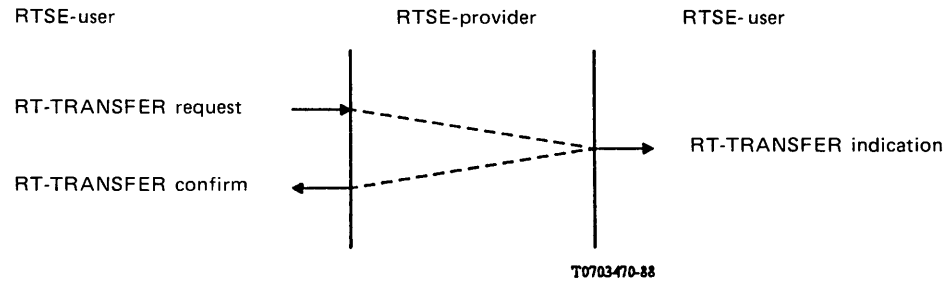


FIGURE 4/X.218

RT-TRANSFER Service-primitives

The RT-TRANSFER confirm primitive signifies that the APDU has been secured by the receiving RTSE-provider (positive confirm), or that the requested transfer of an APDU could not be completed within the specified transfer time (negative confirm).

9.3.1 *RT-TRANSFER parameters*

Table 4/X.218 lists the RT-TRANSFER service parameters.

TABLE 4/X.218

RT-TRANSFER parameters

Parameter name	Request	Indication	Confirmation
APDU	M	M(=)	T(=)
Transfer-time	M		
Result			M

9.3.1.1 *APDU*

This parameter contains the RTSE-user APDU value to be transferred. This parameter has to be supplied by the requestor of the RT-TRANSFER service and, in the case of a negative confirm, by the service-provider.

9.3.1.2 *Transfer-time*

This parameter defines the time period within which the RTSE-provider shall successfully transfer the APDU to the other RTSE-user. This parameter has to be supplied by the requestor of the RT-TRANSFER service.

9.3.1.3 *Result*

This parameter specifies the result of the transfer as follows:

- APDU-transferred: positive confirm; the APDU has been transferred to, and secured by, the receiving RTSE-provider;
- APDU-not-transferred: negative confirm; the APDU could not be transferred within the specified transfer time. *Note* – In certain unusual circumstances a negative confirm may be reported even though the APDU has been transferred to, and secured by, the receiving RTSE-provider.

This parameter has to be supplied by the RTSE-provider.

9.4 *RT-TURN-PLEASE service*

The RT-TURN-PLEASE service enables an RTSE-user to request the Turn. It may do so only if it does not already possess the Turn. The Turn is requested by either RTSE-user to allow the RTSE-user to transfer APDUs. The Turn is requested by the association-initiating RTSE-user to allow it to release the application-association. The request conveys the priority of the action to be taken so that the other RTSE-user can decide when to actually relinquish the Turn. This service is a non-confirmed service.

The related service structure consists of two service-primitives, as illustrated in Figure 5/X.218.

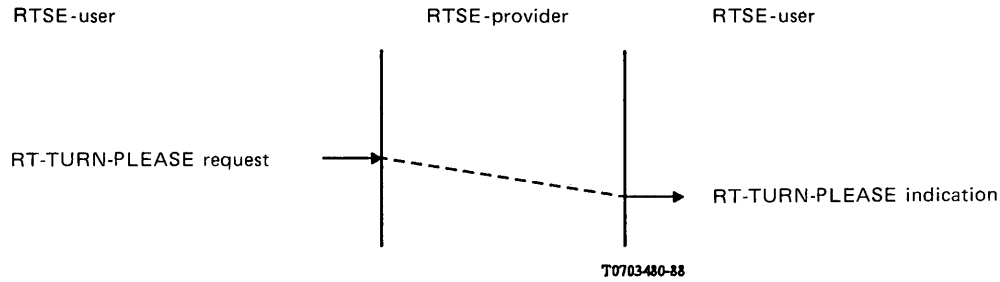


FIGURE 5/X.218

RT-TURN-PLEASE Service-primitives

9.4.1 *RT-TURN-Please parameters*

Table 5/X.218 lists the RT-TURN-PLEASE service parameters.

TABLE 5/X.218

RT-TURN-PLEASE parameters

Parameter name	Request	Indication
Priority	U	C(=)

9.4.1.1 *Priority*

This parameter defines the priority of the action, governed by the Turn, that the requestor of the RT-TURN-PLEASE service wishes to carry out. A priority is assigned to each RTSE-user action. Priority zero is the highest priority and is reserved for the action of releasing an application-association. The actions of transferring various APDUs will be assigned other priorities. The range of valid priorities is a property of the application context in use. This parameter has to be supplied by the requestor of the RT-TURN-PLEASE service.

If the Priority parameter is absent, priority zero is assumed.

9.5 *RT-TURN-GIVE service*

The RT-TURN-GIVE service enables an RTSE-user to relinquish the Turn to its peer. It may do so only if it possesses the Turn and there is no outstanding RT-TRANSFER confirm primitive. This service is a non-confirmed service.

The related service structure consists of two service-primitives, as illustrated in Figure 6/X.218.

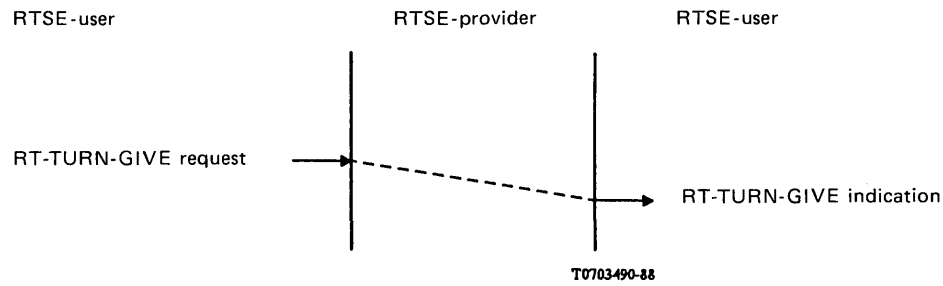


FIGURE 6/X.218

RT-TURN-GIVE Service-primitives

9.5.1 *RT-TURN-GIVE parameters*

The RT-TURN-GIVE service has no parameters.

9.6 *RT-P-ABORT service*

The RT-P-ABORT services provides an indication to both the RTSE-users that the application-association cannot be maintained (e.g., because recovery not possible, etc.). If it is the sender, the RTSE-provider first issues a negative RT-TRANSFER confirm primitive for the APDU not yet transferred. If it is the receiver, the RTSE-provider deletes any partially received APDUs prior to issuing the RT-P-ABORT indication. This service is a provider-initiated service.

The related service structure consists of two service-primitives, as illustrated in Figure 7/X.218.

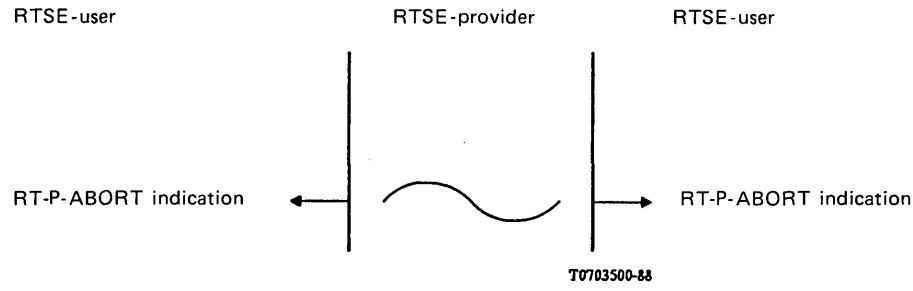


FIGURE 7/X.218

RT-P-ABORT Service-primitives

9.6.1 *RT-P-ABORT parameters*

The RT-P-ABORT service has no parameters.

9.7 *RT-U-ABORT service*

The RT-U-ABORT service enables an RTSE-user to abort the application-association. The abort may be requested by either RTSE-user. This service is a non-confirmed service.

Note – This service is not supported in X.410-1984 mode.

The related service structure consists of two service-primitives, as illustrated in Figure 8/X.218.

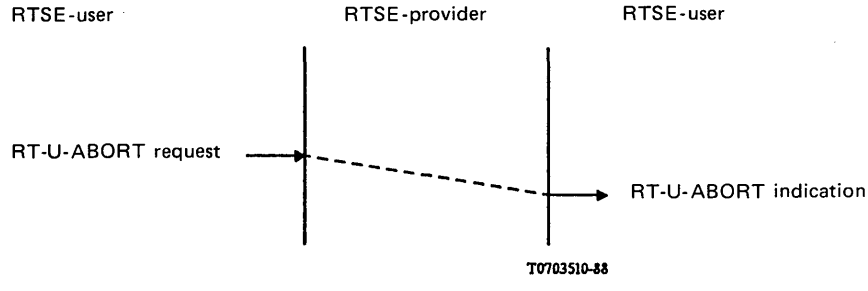


FIGURE 8/X.218

RT-U-ABORT Service-primitives

9.7.1 *RT-U-ABORT parameters*

Table 6/X.218 lists the RT-U-ABORT service parameters.

TABLE 6/X.218

RT-U-ABORT parameters

Parameter name	Request	Indication
User-data	U	C(=)

9.7.1.1 *User-data*

User-data associated with aborting the application-association. This parameter has to be supplied by the requestor of the RT-U-ABORT service.

10 Sequencing information

This clause defines the interaction among the RTSE services.

10.1 *RT-OPEN*

10.1.1 *Type of service*

The RT-OPEN service is a confirmed service.

10.1.2 *Usage restrictions*

The RT-OPEN service is not used on an established application-association.

10.1.3 *Disrupted services*

The RT-OPEN service does not disrupt any service.

10.1.4 *Disrupting services*

The RT-OPEN service is disrupted by the RT-P-ABORT service and the RT-U-ABORT service.

10.1.5 *Collisions*

An RT-OPEN collision results when requestors in both AEs simultaneously issue an RT-OPEN request primitive for each other. In this case two independent application-associations are established.

10.2 *RT-CLOSE*

10.2.1 *Type of service*

The RT-CLOSE service is a confirmed service.

10.2.2 *Usage restrictions*

The RT-CLOSE service is used only on an established application-association by the association-initiator. It is only used when the association-initiator possesses the Turn, and when there is no outstanding RT-TRANSFER confirm primitive.

10.2.3 *Disrupted services*

The RT-CLOSE service does not disrupt any service.

10.2.4 *Disrupting services*

The RT-CLOSE service is disrupted by the RT-P-ABORT service and the RT-U-ABORT service.

10.2.5 *Collisions*

Because only the association-initiator uses this service, there is no collision of the RT-CLOSE service.

10.3 *RT-TRANSFER service*

10.3.1 *Type of service*

The RT-TRANSFER service is a confirmed service.

10.3.2 *Usage restriction*

The RT-TRANSFER service is only used on an established application-association, and if the RTSE-user possesses the Turn, and if there is no outstanding RT-TRANSFER confirm primitive.

10.3.3 *Disrupted services*

The RT-TRANSFER service does not disrupt any services.

10.3.4 *Disrupting services*

The RT-TRANSFER service is disrupted by the RT-P-ABORT service and the RT-U-ABORT service, in the sense that a negative RT-TRANSFER confirm primitive and no RT-TRANSFER indication primitive may occur.

10.3.5 *Collision*

There is no collision of RT-TRANSFER services.

10.4 *RT-TURN-PLEASE service*

10.4.1 *Type of service*

The RT-TURN-PLEASE service is a non-confirmed service.

10.4.2 *Usage restriction*

The RT-TURN-PLEASE service is only used on an established application-association, and if the RTSE-user does not already possess the Turn.

10.4.3 *Disrupted services*

The RT-TURN-PLEASE service does not disrupt any services.

10.4.4 *Disrupting services*

The RT-RETURN-PLEASE service is disrupted by the RT-P-ABORT service and the RT-U-ABORT service.

10.4.5 *Collision*

There is no collision of RT-TURN-PLEASE services.

10.5 *RT-TURN-GIVE service*

10.5.1 *Type of service*

The RT-TURN-GIVE service is a non-confirmed service.

10.5.2 *Usage restriction*

The RT-TURN-GIVE service is only used on an established application-association, and if the RTSE-user possesses the Turn, and if there is no outstanding RT-TRANSFER confirm primitive.

10.5.3 *Disrupted services*

The RT-TURN-GIVE service does not disrupt any services.

10.5.4 *Disrupting services*

The RT-TURN-GIVE service is disrupted by the RT-P-ABORT service and the RT-U-ABORT service.

10.5.5 *Collision*

There is no collision of RT-TURN-GIVE services.

10.6 *RT-P-ABORT service*

10.6.1 *Type of service*

The RT-P-ABORT service is provider initiated service.

10.6.2 *Usage restriction*

Not applicable.

10.6.3 *Disrupted services*

The RT-P-ABORT service disrupts all other RTSE services.

10.6.4 *Disrupting services*

The RT-P-ABORT service is not disrupted by any other service.

10.6.5 *Collision*

If the RT-P-ABORT service causes an abort of an application-association, it is a local matter to inform the service-user about the outstanding negative RT-TRANSFER confirm primitive for an APDU not transferred.

10.7 *RT-U-ABORT service*

10.7.1 *Type of service*

The RT-U-ABORT service is a non-confirmed service.

10.7.2 *Usage restriction*

The RT-U-ABORT service is only used on an established application-association.

10.7.3 *Disrupted services*

The RT-U-ABORT service disrupts all other RTSE services, except the RT-P-ABORT service.

10.7.4 *Disrupting services*

The RT-U-ABORT service is disrupted by the RT-P-ABORT service.

10.7.5 *Collision*

If the RT-U-ABORT service causes an abort of an application-association, it is a local matter to inform the service-user about the outstanding negative RT-TRANSFER confirm primitive for an APDU not transferred.

REMOTE OPERATIONS: MODEL, NOTATION AND SERVICE DEFINITION[1)]

(Melbourne, 1988)

The CCITT,

considering

(a) that Recommendation X.200 defines the 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.229 defines the Remote Operations protocol;

(g) that there is a need for common Remote Operations support for various applications;

unanimously declares

that this Recommendation defines the Remote Operation service and notation of Open Systems Interconnection for CCITT Applications as given in the Scope and Field of Application.

CONTENTS

[1)] Recommendation X.219 and ISO 9072-1 [Information Procesing Systems – Text Communications – Remote Operations, Part 1: Model, Notation and Service Definition] were developed in close collaboration and are technically aligned

Annex A – Notation Supporting the Specification of application-service-elements and Application Contexts

Annex B – Guidelines for application Protocol Designers on the Use of ROSE

0 Introduction

This Recommendation defines a notation and 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 9072-1.

1 Scope and Field of Application

This Recommendation defines a Remote Operation (RO-) Notation for defining the services provided to interactive applications. This Recommendation also defines the services provided by the Remote Operation Service Element (ROSE) services. The ROSE services are provided by the use of the ROSE protocol (Recommendation X.229) 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).

No requirement is made for conformance to this Recommendation.

2 References

X.200 Reference Model of Open Systems Interconnection for CCITT Applications (see also ISO 7498).

X.208 Specification of abstract syntax notation (see also ISO 8824).

X.209 Specification of Basic Encoding Rules for the abstract syntax notation (see also ISO 8825).

X.210 Open Systems Interconnection Layer Service Definition Conventions (see also ISO/TR 8509).

X.216 Presentation Service Definition for Open Systems Interconnection for CCITT applications (see also ISO 8822).

X.217 Association Control Service Definition for CCITT Applications (see also ISO 8649).

X.218 Reliable Transfer: Model and Service Definition (see also ISO 9066-1).

X.227 Association Control Protocol Specification for CCITT Applications (see also ISO 8650).

X.228 Reliable Transfer: Protocol Specification (see also ISO 9066-2).

X.229 Remote Operations: Protocol Specification (see also ISO 9072-2).

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 Layer;
h) presentation-service;
i) presentation-connection;
j) session-service;
k) session-connection;
l) transfer syntax; and
m) 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) service-primitive; 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) transfer syntax name;
d) 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;

3.5 *Reliable Transfer Definitions*

This Recommendation makes use of the following terms defined in Recommendation X.218:
a) Reliable Transfer Service Element.

3.6 ROSE *Definitions*

For the purpose of this Recommendation the following definitions apply:

3.6.1 **association-initiating-application-entity; association-initiator**

The application-entity that initiates the application-association.

3.6.2 **association-responding-application-entity; association-responder**

The application-entity that responds to the initiation of an application-association by another AE.

3.6.3 **invoking-application-entity; invoker**

The application-entity that invokes the Remote Operation.

3.6.4 **performing-application-entity ; performer**

The application-entity that performs a Remote Operation invoked by the other application-entity.

3.6.5 **requestor**

The part of an application-entity that issues a request primitive for a particular ROSE service.

3.6.6 **acceptor**

The part of an application-entity that receives the indication primitive for a particular ROSE service.

3.6.7 **linked-operations**

A set of operations formed by one parent-operation and one or more child-operations.

3.6.8 **parent-operation**

An operation during the execution of which the performer may invoke linked child-operations to be performed by the invoker of the parent-operation.

3.6.9 **child-operation**

An operation which might be invoked by the performer of the linked parent-operation during the execution of the parent-operation, and which is performed by the invoker of the parent-operation.

3.6.10 **Remote Operations**

1) A concept and notation supporting the specification of interactive communication between application-entities. This includes the Remote Operation Service Element and the mapping of the notation onto the service primitives of used application-service-elements.
2) The set of bind-operations, unbind-operations and operations.

3.6.11 **RO-notation**

The notation used for the specification of Remote Operations, defined in this Recommendation.

3.6.12 **ACSE-user**

The application-specific function that performs the mapping of the bind-operation and unbind-operation of the RO-notation onto ACSE.

3.6.13 **Remote Operation Service Element**

The application-service-element defined in this Recommendation.

3.6.14 **ROSE-provider**

The provider of the Remote Operations Service Element services.

3.6.15 **ROSE-user**

The application-specific function that performs the mapping of the operations and errors of the RO-notation onto ROSE.

3.6.16 **RTSE-user**

The application-specific function that performs the mapping of the bind-operation and unbind-operation of the RO-notation onto RTSE.

3.6.17 **operation-interface**

The interface within an application entity between the user element and the application service elements, defined as a set of application service element services (Remote Operations) available to the user element in RO-notation.

4 Abbreviations

AE	application-entity
ACSE	Association Control Service Element
ASE	application-service-element
APDU	application-protocol-data-unit
OSI	Open Systems Interconnection
RO (or ROS)	Remote Operations
ROSE	Remote Operations Service Element
RT (or RTS)	Reliable Transfer
RTSE	Reliable Transfer Service Element

5 Conventions

This Recommendation defines services for the ROSE following the descriptive conventions defined in Recommendation X.210. In § 10, the definition of each ROSE service includes a table that lists the parameters of its primitives. For a given primitive, the presence of each parameter is decribed by one of the following values:

blank not applicable

M	mandatory
U	user option
C	conditional
O	presence is a ROSE service-provider option

In addition, the notation (=) indicates that a parameter value is semantically equal to the value to its left in the table.

6 Remote Operations Model

In the OSI environment, communication between application processes is represented in terms of communication between a pair of application entities (AEs) using the presentation service. Communication between some application-entities are inherently interactive. Typically, one entity requests that a particular operation be performed; the other entity attempts to perform the operation and then report the outcome of the attempt. This Section introduces the concept of Remote Operations as a vehicle for supporting interactive applications.

The generic structure of an operation is an elementary request/reply interaction. Operations are carried out within the context of an application-association.

Figure 1/X.219 models this view.

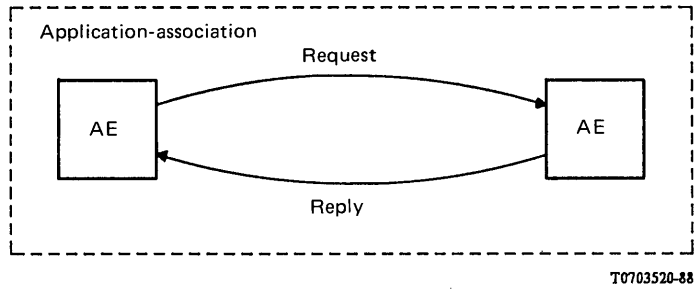


FIGURE 1/X.219

Remote operations model

Operations invoked by one AE (the invoker) are performed by the other AE (the performer). Operations may be classified according to whether the performer of an operation is expected to report its outcome:

- in case of success or failure (a result reply is returned if the operation is successful, an error reply is returned if the operation is unsuccessful);
- in case of failure only (no reply is returned if the operation is successful, an error reply is returned if the operation is unsuccessful);
- in case of success only (a result reply is returned if the operation is successful, no reply is returned if the operation is unsuccessful);
- or not at all (neither a result nor an error reply is returned, whether the operation was successful or not).

Operations may also be classified according to two possible operation modes: synchronous, in which the invoker requires a reply from the performer before invoking another operation; an asynchronous, in which the invoker may continue to invoke further operations without awaiting a reply.

The following Operation Classes are defined:

Operation Class 1: Synchronous, reporting success or failure (result or error).

Operation Class 2: Asynchronous, reporting success or failure (result or error).

Operation Class 3: Asynchronous, reporting failure (error) only, if any.

Operation Class 4: Asynchronous, reporting success (result) only.

Operation Class 5: Asynchronous, outcome not reported.

The Operation Class of each operation has to be agreed between application entities (e.g. in an Application Protocol Recommendation).

In some cases it is useful to group operations into a set of linked-operations which is formed by one parent-operation and one or more child-operations. The performer of the parent-operation may invoke none, one, or more child-operations during the execution of the parent-operation. The invoker of the parent-operation is the performer of the child-operations. A child-operation may be a parent-operation of another set of linked-operations in a recursive manner. Figure 2/X.219 models this concept.

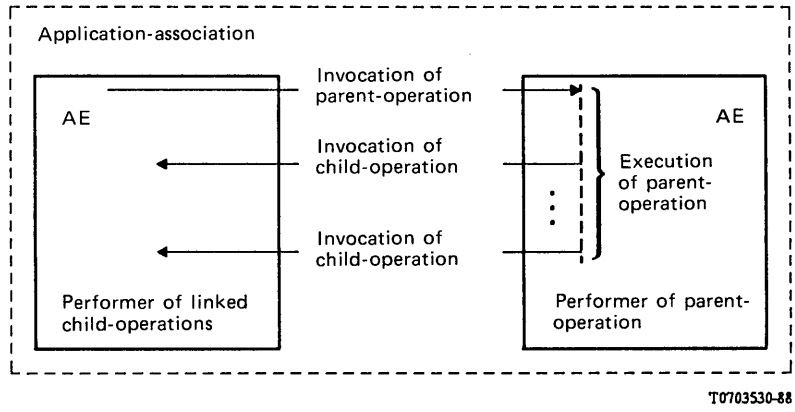


FIGURE 2/X.219

Linked-operations

An application-association defines the relationship between a pair of AEs, and is formed by the exchange of application-protocol-control-information through the use of presentation-services. The AE that initiates an application-association is called the association-initiating AE, or the association-initiator, while the AE that responds to the initiation of an application-association by another AE is called the association-responding AE, or the association-responder. Only the association-initiating AE may release an established application-association.

Application-associations are classified by which application-entity is allowed to invoke operations:

Association Class 1: Only the association-initiating application entity can invoke operations.

Association Class 2: Only the association-responding application entity can invoke operations.

Association Class 3: Both the association-initiating and the association-responding application entities can invoke operations.

Linked-operations require Association Class 3.

The Association Class has to be agreed between application-entities (e.g. in an Application Protocol Recommendation).

The functionality of an AE is factored into one user-element and a set of application-service-elements (ASEs). Each ASE may itself be factored into a set of (more primitive) ASEs. The interaction between AEs is described in terms of their use of ASEs.

The specific combination of a user-element and the set of ASEs which comprise an AE defines the application-context.

Figure 3/X.219 illustrates an example of an application-context involving the Remote Operations Service Element (ROSE). Note that this figure is not meant to imply that the application is symmetric. Interactive applications are often inherently asymmetric, that is, either one or both AEs may be permitted to invoke operations, and the operations that either AE may invoke may be different. The rules governing which AE may invoke operations, and which operations an AE may invoke, is defined using the RO-notation in an Application Protocol Recommendation, and determines the application-context.

The set of ASEs available to the user element of the AE at the operation-interface is defined using the Remote Operations (RO-) Notation. The RO-notation is based on the macro concept defined in Recommendation X.208. The complexity of a particular set of ASEs is dependent upon the needs of the application, and is not limited by the Remote Operations concept.

An important characteristic of Remote Operations is that they provide applications with independence from OSI communication services. Since the notation is based on established object-oriented programming principles, automatic tools can be developed to bind Remote Operations into the execution environment of applications.

The ASEs available to the user-element require communication over an application-association. The control of that application-association (establishment, release, abort) is performed either by the Association Control Service Element (ACSE) defined in Recommendation X.217, or the Reliable Transfer Service Element defined in Recommendation X.218 and the Association Control Service Element (ACSE). Communication over the application-association is performed by the Remote Operations Service Element (ROSE) defined in this Recommendation.

An application-specific function performs the mapping of the operations available to the user-element onto either the ACSE services, or the RTSE services; and the ROSE services. The mapping is defined in this Recommendation. The function that performs the mapping of the operations onto the ACSE services, or the RTSE services, and the ROSE services is said to be the user of ACSE, RTSE and ROSE, or the ACSE-user, the RTSE-user, and the ROSE-user.

If the RTSE is included in the application-context, the mapping function is an RTSE-user and a ROSE-user, the ROSE is an RTSE-user, the RTSE is an ACSE-user and a presentation service-user, and the ACSE is a presentation service-user.

If the RTSE is excluded from the application-context, the mapping function is an ACSE-user and a ROSE-user, the ROSE is a presentation service-user, and the ACSE is a presentation service-user.

7 Overview of Notation and Service

7.1 *Notation Overview*

This Recommendation defines the RO-notation for the specification of an application-context and the related abstract syntax component of the presentation context.

The functionality of an application-context is provided to the user-element by means of Remote Operations and errors which form the operation-interface.

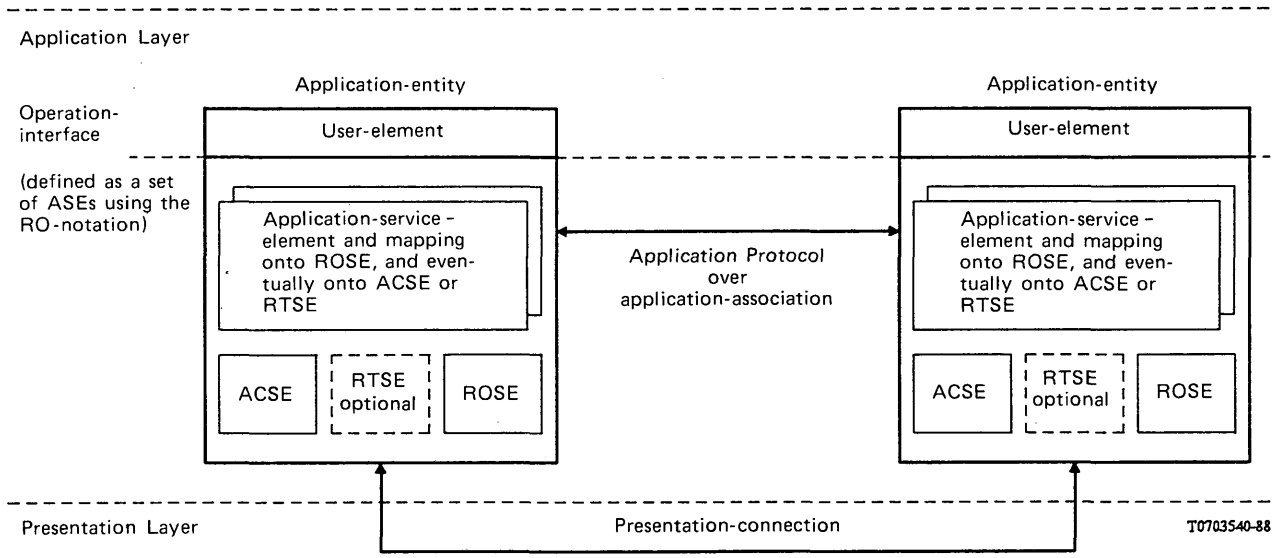


FIGURE 3/X.219

Model of an application context involving remote operations

The following types of Remote Operations form an operation interface:

- a bind-operation to establish an application-association
- a set of operations and, for each operation, a list of error (negative reply) situations
- an unbind-operation to release an application-association.

The abstract syntax notation of Recommendation X.208 is used for the definition of the following macros:

a) BIND;

b) UNBIND;

c) OPERATION; and

d) ERROR.

These macros provide both a type notation and a value notation for Remote Operations and errors.

The type notation of the BIND macro enables the specification for a bind-operation type and the types for user data values (if any) to be exchanged in the establishment phase of an application-assocation. The value notation of the BIND macro enables the specification of user data values (if any) to be exchanged in the establishment phase of an application-association.

The type notation of the UNBIND macro enables the specification of an unbind-operation type and types for user data values (if any) to be exchanged in the release phase of an application association. The value notation of the UNBIND macro enables the specification of user data values (if any) to be exchanged in the release phase of an application-association.

The type notation of the OPERATION macro enables the specification of an operation and user data types to be exchanged for a request and a positive reply. In addition, the type notation enables the specification of a list of valid negative reply situations. If the operation is a parent-operation, the type notation enables the specification of the list of linked child-operations. The value notation of the OPERATION macro enables the specification of the identifier of an operation.

The type notation of the ERROR macro enables the specification of user data types to be exchanged in a negative reply situation. The value notation of the ERROR macro enables the specification of the identifier of an error.

Additional macros supporting the notation for the specification of application-service-elements and application context are defined in Annex A.

7.2 *Service Overview*

This Recommendation defines the following ROSE services:

a) RO-INVOKE

b) RO-RESULT

c) RO-ERROR

d) RO-REJECT-U

e) RO-REJECT-P

The RO-INVOKE service enables an invoking AE to request an operation to be performed by the performing AE.

The RO-RESULT service enables the performing AE to return the positive reply of a successfully performed operation to the invoking AE.

The RO-ERROR service enables the performing AE to return the negative reply of an unsuccessfully performed operation to the invoking AE.

The RO-REJECT-U service enables one AE to reject the request or reply of the other AE if the ROSE-user has detected a problem.

The RO-REJECT-P service enables the ROSE-user to be informed about a problem detected by the ROSE-provider.

7.3 *Mapping of Notation onto Services*

Note that the function that performs the mapping of the OPERATION macros and ERROR macros of the RO-notation onto ROSE services is said to be the ROSE-user. While the function that performs the mapping of the BIND and UNBIND macros of the RO-notation onto ACSE services or RTSE services respectively is said to be the ACSE-user or RTSE-user respectively.

The specification of the mapping of the RO-notation onto the used services of ACSE, RTSE, and ROSE is given in § 11. Therefore Recommendations using the RO-notation for the protocol specification need not specify the mapping onto these used services.

8 Relationship with other ASEs and Lower Layer Services

8.1 *Other Application Service Elements*

The ROSE is intended to be used with other ASEs in order to support specific interactive information processing tasks. Therefore it is expected that the ROSE will be included in a large number of application-context specifications.

The collection of the ROSE and other ASEs included in an application context are required to use the facilities of the presentation-service in a co-ordinated manner among themselves.

The ROSE requires an existing application-association controlled by ACSE.

For some application context specifications a Reliable Transfer Service Element (RTSE) is included.

A ROSE-user protocol specification uses the RO-notation. It defines one or more abstract syntaxes and provides unique abstract syntax names of type object identifier for each abstract syntax.

If a named abstract syntax specifies operations and errors, the ROSE APDUs defined in Recommendation X.229 are included in that named abstract syntax. If multiple named abstract syntaxes are defined for operations and errors, the ROSE APDUs are included in each named abstract syntax.

If a named abstract syntax specifies a bind-operation, the APDUs specified by the value notation of the BIND macro are included in that named abstract syntax. If the RTSE is included in the application context, the APDUs for the bind-operation share a single named abstract syntax with the RTSE APDUs defined in Recommendation X.228.

If a named abstract syntax specifies an unbind-operation, the APDUs specified by the value notation of the UNBIND macro are included in that named abstract syntax.

The APDUs resulting from the specification of a bind-operation, an unbind-operation, operations and errors and the RTSE APDUs may share a single named abstract syntax.

8.2 *Presentation-Service*

If an application context including RTSE and ROSE is defined, ROSE services do not use the presentation-service.

If an application context including ROSE but excluding RTSE is defined, the ROSE services require access to the P-DATA service and require the use of the duplex functional unit of the presentation-service. The ROSE services neither use, nor constrain the use of, any other presentation service.

A named abstract syntax associated with a compatible transfer syntax (negotiated by the Presentation Layer) constitutes a presentation context.

The object identifier value {joint-iso-ccitt ans1(1) basic-encoding(1)} specified in Recommendation X.209 may be used as a transfer syntax name. In this case the ROSE-user protocol specification need not name nor specify a transfer syntax.

9 Remote Operations Notation

9.1 *General*

The notation used in this Recommendation is defined as follows:

- the data syntax notation and macro notation are defined in Recommendation X.208;
- the Remote Operation macros are defined in § 9.2 of this Recommendation.

A bind-operation defines where an object binding (establishment of an application-association) begins. If such a binding is established, operations may be invoked. An unbind-operation defines where an object binding is released.

An interactive protocol is specified using the Remote Operation and error data types. This section defines those types. It also explains the notational definitions of a particular Remote operation, and of the particular errors it can report. The notation is defined by means of the macro facility defined in Recommendation X.208. This macro definition allows a generalized specification of the mapping onto various execution environments.

The macros enabling the specification of bind-operations, unbind-operations, operations and errors are listed in Figure 4/X.219.

9.2 *Specification of Bind-operations*

A single data value, the argument of the bind-operation, may accompany the request to establish the application-association. Some bind-operations report their outcome, whether success (i.e. the normal outcome) or failure (i.e. the exceptional outcome). Other bind-operations report their outcome only if they fail, and still others never at all. A single data value, the result of the bind-operation, may accompany the positive response. A single data value, the bind-error of the bind-operation, may accompany the negative response.

The notation for a bind-operation type is the keyworkd BIND, optionally followed by the keyword ARGUMENT and the type of the bind-operation's argument, the reference name optionally assigned to it, and the nature of the operation's outcome reporting (if any). If the bind-operation reports success, the keyword RESULT and the type of its result and the reference name optionally assigned to it are specified. If the bind-operation reports failure, the keyword BIND-ERROR and the type of the error-information it reports and the reference name optionally assigned to it are specified.

The value notation for a bind-operation is either an argument value, or a result value or an error value. The value notation for an argument value (if any) is the keyword ARGUMENT followed by a value of the argument type. The value notation for a result value (if any) is the keyword RESULT followed by a value of the result type. The value notation for an error value (if any) is the keyword ERROR followed by a value of the error type.

```
Remote-Operation-Notation {joint-iso-ccitt remote-operations(4) notation(0)}
DEFINITIONS ::=
BEGIN

EXPORTS BIND, UNBIND, OPERATION, ERROR;

-- macro definition for bind-operations

BIND MACRO ::=
BEGIN

TYPE NOTATION          ::= Argument Result Error
VALUE NOTATION         ::= Argument-value | Result-value | Error-value

Argument               ::= empty |"ARGUMENT" Name type (Argument-type)
                           -- Expects any ASN.1 type and assigns it to the variable Argument-type

Result                 ::= empty |"RESULT" Name type (Result-type)
                           -- Expects any ASN.1 type and assigns it to the variable Result-type

Error                  ::= empty |"BIND-ERROR" Name type (Error-type)
                           -- Expects any ASN.1 type and assigns it to the variable Error-type

Name                   ::= empty | identifier

Argument-value         ::= empty |"ARGUMENT" value (Arg-value Argument-type)
                           -- Expects a value for the type in Argument-type, and assigns it to the
                           -- variale Arg-value
                           <VALUE [16] EXPLICIT Argument-type ::= Arg-value>
                           -- Returns the final value as explicitly tagged type

Result-value           ::= empty |"RESULT" value (Res-value Result-type)
                           -- Expects a value for the type in Result-type, and assigns it to the
                           -- variable Res-value
                           <VALUE [17] EXPLICIT Result-type ::= Res-value>
                           -- Returns the final value as explicitly tagged type

Error-value            ::= empty |"ERROR" value (Err-value Error-type)
                           -- Expects a value for the type in Error-type, and assigns it to the
                           -- variable Err-value
                           <VALUE [18] EXPLICIT Error-type ::= Err-value>
                           -- Returns the final value as explicitly tagged type

END
-- Remote Operations Notation continued
```

FIGURE 4/X.219 (Part 1 of 3)

Formal Definition of Remote Operations Data Types

```
-- Remote Operations Notation continued

-- macro definition for unbind-operations

UNBIND MACRO ::=

BEGIN

TYPE NOTATION          ::= Argument Result Errors
VALUE NOTATION         ::= Argument-value | Result-value | Error-value

Argument               ::= empty |"ARGUMENT" Name type (Argument-type)
                               -- Expects any ASN.1 type and assigns it to the variable Argument-type

Result                 ::= empty |"RESULT" Name type (Result-type)
                               -- Expects any ASN.1 type and assigns it to the Result-type

Error                  ::= empty |"UNBIND-ERROR" Name type (Error-type)
                               -- Expects any ASN.1 type and assigns it to the Error-type

Name                   ::= empty | identifier

Argument-value         ::= empty |"ARGUMENT" value (Arg-value Argument-type)
                               -- Expects a value for the type in Argument-type, and assigns it to the
                               -- variable Arg-value
                               <VALUE [19] EXPLICIT Argument-type ::= Arg-value>
                               -- Returns the final value as explicitly tagged type

Result-value           ::= empty |"RESULT" value (Res-value Result-type)
                               -- Expects a value for the type in Result-type and assigns it to the
                               -- variable Res-value
                               <VALUE [20] EXPLICIT Result-type ::= Res-value>
                               -- Returns the final value as explicitly tagged type

Error-value            ::= empty |"ERROR" value (Err-value Error-type)
                               -- Expects a value for the type in Error-type and assigns it to the
                               -- variable Err-value
                               <VALUE [21] EXPLICIT Error-type ::= Err-value>
                               -- Returns the final value as explicitly tagged type

END

-- Remote Operations Notation continued
```

FIGURE 4/X.219 (Part 2 of 3)

Formal Definition of Remote Operations Data Types

```
-- Remote Operations Notation continued
-- macro definition for operations
OPERATION MACRO ::=
BEGIN
TYPE NOTATION       ::=  ArgumentResultErrorsLinkedOperations
VALUE NOTATION      ::=  value (VALUE CHOICE {
                                    localValue   INTEGER,
                                    globalValue OBJECT IDENTIFIER})
Argument            ::=  "ARGUMENT" NamedType | empty
Result              ::=  "RESULT" ResultType | empty
ResultType          ::=  NamedType | empty
Errors              ::=  "ERRORS" "{"ErrorNames"}" | empty
LinkedOperations    ::=  "LINKED" "{"LinkedOperationsNames"}" | empty
ErrorNames          ::=  ErrorList | empty
ErrorList           ::=  Error | ErrorList "," Error
Error               ::=  value (ERROR)          -- shall reference an error value
                         | type    -- shall reference an error type if no error value is specified
LinkedOperation-    ::=  OperationList | empty
Names
OperationList       ::=  Operation | OperationList "," Operation
Operation           ::=  value (OPERATION)      -- shall reference an operation value
                         | type    -- shall reference an operation type if no operation value is specified
NamedType           ::=  identifier type | type
END
-- macro definition for operations errors
ERROR MACRO ::=
BEGIN
TYPE NOTATION       ::=  Parameter
VALUE NOTATION      ::=  value (VALUE CHOICE {
                                    localValue   INTEGER,
                                    globalValue OBJECT IDENTIFIER})
Parameter           ::=  "PARAMETER" NamedType | empty
NamedType           ::=  identifier type | type
END
END     -- end of Remote Operations Notation
```

FIGURE 4/X.219 (Part 3 of 3)

Formal Definition of Remote Operations Data Types

9.3 *Specifications of Unbind-operations*

A single data value, the argument of the unbind-operation, may accompany the request to release the application-association. Some unbind-operations report their outcome, whether success (i.e. the normal outcome) or failure (i.e. the exceptional outcome). Other unbind-operations report their outcome only if they fail, and still others never at all. A single data value, the result of the unbind-operation, or a single data value, the unbind-error of the unbind-operation, may accompany the response.

The notation for an unbind-operation type is the keyword UNBIND, optionally followed by the keyword ARGUMENT and the type of the unbind-operation's argument, the reference name optionally assigned to it, and the nature of the unbind-operation's outcome reporting (if any). If the unbind-operation reports success, the keyword RESULT and the type of its result and the reference name optionally assigned to it are specified. If the Unbind-operation reports failure, the keyword UNBIND-ERROR and the type of the error-information it reports and the reference name optionally assigned to it are specified.

The value notation for an unbind-operation is either an argument value, or a result value or an error value. The value notation for an argument value (if any) is the keyword ARGUMENT followed by a value of the argument type. The value notation for a result value (if any) is the keyword RESULT followed by a value of the result type. The value notation for an error value (if any) is the keyword ERROR followed by a value of the error type.

9.4 *Specification of Operations*

A data value of type operation represents the identifier for an operation that a ROSE-user in one open system may request to be performed by a peer ROSE-user in another open system. A single data value, the argument of the operation, may accompany the request. Some operations report their outcome, whether success (i.e. the normal outcome) or failure (i.e. the exceptional outcome). Other operations report their outcome only if they fail, and still others never at all. A single data value, the result of the operation, accompanies a report of succes; a report of failure identifies the exceptional condition that was encountered.

The notation for an operation type is the keyword OPERATION, optionally followed by the keyword ARGUMENT and the type of the operation's argument, the reference name optionally assigned to it, and the nature of the operation's outcome reporting (if any). If the operation reports success, the keyword RESULT and optionally the type of its result and the reference name optionally assigned to it are specified. If the operation reports failure, the keyword ERRORS and the reference names of the error values or error types it reports are specified. If the operation is the parent-operation of a set of linked-operations, the keyword LINKED-OPERATIONS and the reference names of the linked child-operation values or child-operation types are specified. The reference to error values or child-operation values is preferred, however, the references to types shall be used if the values are defined elsewhere (see § 9.6).

The notation for an operation value is the operation's identifier. If a locally unique identifier (local value) is sufficient, the identifier is of type INTEGER. If a globally unique identifier (global value) is required to allow the unique identification of operations used in several abstract syntaxes, the identifier is of type OBJECT IDENTIFIER.

Child-operations and errors referenced by a specific operation shall share a single named abstract syntax (see sub- § 8.1) with that operation, if the child-operations or errors are identified by local values. The use of global values is not restricted.

9.5 *Specification of Errors*

A data value of type error represents the identifier for an exception condition that a ROSE-user in one open system may report to a peer ROSE-user in another open system, where the exception condition is reporting an exceptional outcome of a previously requested operation. A single data value, the parameter of the error, may accompany the report.

The notation for an error type is the keyword ERROR, optionally followed by the keyword PARAMETER and the type of the error's parameter and the reference name optionally assigned to it.

The notation for an error value is the error's identifier. If a locally unique identifier (local value) is sufficient, the identifier is of type INTEGER. If a globally unique identifier (global value) is required to allow the unique identification of errors used in several abstract syntaxes, the identifier is of type OBJECT IDENTIFIER.

9.6 *Export and Import of Operations and Errors*

Operation values and error values have to be unique within a named abstract syntax. If operations and errors are specified in several ASN.1 modules and are imported to a module specifying a specific named abstract syntax, one of the following rules apply:

1) If local values are used and exported, it is in the responsibility of the designer of the importing module to ensure uniqueness.
2) A module may specify and export operation types and error types. The operation values and error values are assigned in the module importing the types. A single value shall be assigned for each operation type or error type.
3) If global values are assigned and exported, uniqueness is ensured.

However different named abstract syntaxes might be used for conflicting local values.

10 Service Definition

The ROSE services are listed in Table 1/X.219.

TABLE 1/X.219

ROSE Services

Service	Type
RO-INVOKE	Non-confirmed
RO-RESULT	Non-confirmed
RO-ERROR	Non-confirmed
RO-REJECT-U	Non-confirmed
RO-REJECT-P	Provider-initiated

Identification of the named abstract syntax in use is assumed for all ROSE services, however this is a local matter and outside the scope of this Recommendation.

10.1 *RO-INVOKE Service*

The RO-INVOKE service is used by one ROSE-user (the invoker) to cause the invocation of an OPERATION to be performed by the other ROSE-user (the performer). This service is a non-confirmed service.

The related service structure consists of two service-primitives, as illustrated in Figure 5/X.219.

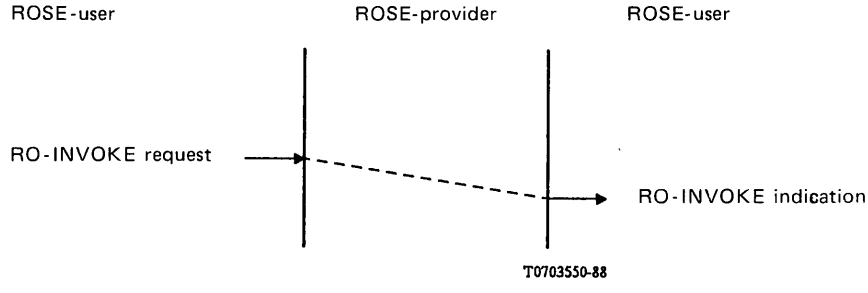


FIGURE 5/X.219

RO-INVOKE Service-primitives

10.1.1 *RO-INVOKE Parameters*

Table 2/X.219 lists the RO-INVOKE service parameters.

TABLE 2/X.219

RO-INVOKE Parameters

Parameter name	Request	Indication
Operation-value	M	M(=)
Operation-class	U	
Argument	U	C(=)
Invoke-ID	M	M(=)
Linked-ID	U	C(=)
Priority	U	

10.1.1.1 *Operation-value*

This parameter is the identifier of the operation to be invoked. The value has to be agreed between the ROSE-users. This parameter has to be supplied by the requestor of the service.

10.1.1.2 *Operation Class*

This parameter defines whether a synchronous or an asynchronous reply is expected and the nature of the expected reply, i.e. result and/or error or non (see § 6). This parameter has to be supplied by the requestor of the service. This parameter is used solely to optimize the turn management (see § 8.1.1 of Recommendation X.229).

10.1.1.3 *Argument*

This parameter is the argument of the invoked operation. The type has to be agreed between the ROSE-users. This parameter has to be supplied by the requestor of the service.

10.1.1.4 *Invoke-ID*

This parameter identifies the request of a RO-INVOKE service and is used to correlate this request with the corresponding replies (RO-RESULT, RO-ERROR, RO-REJECT-U, and RO-REJECT-P services) or the invocation of linked child-operations (RO-INVOKE). This paramter has to be supplied by the requestor of the service.

This parameter distinguishes several requests of the service the requestor may have in progress (asynchronous operations). The requestor may begin to reuse Invoke-ID values whenever it chooses, subject to the constraint that it may not reuse an Invoke-ID value that was previously assigned to a request of the service for which it expects, but has not yet received, a reply or the invocation of a linked child-operation.

The ROSE-user to which an RO-INVOKE indication is issued, assumes that an Invoke-ID value violating the above rule is a duplicate; and therefore, it does not perform the invoked operation. Instead, it rejects the duplicate invocation.

If Operation Classes 3, 4 or 5 are used, the requestor of this service may reuse an Invoke-ID value after a reasonably long period of time, or if the reply is carried by other means (e.g. result of a have-you-finished operation).

In some application contexts peer ROSE-users may communicate Invoke-ID values. To support this the type of the Invoke-ID parameter is exported by the module defining the abstract syntax of Remote Operations in § 9 of Recommendation X.229.

10.1.1.5 *Linked-ID*

If this parameter is present, the invoked operation is a child-operatin and the parameter identifies the invocation of the linked parent-operation. This parameter has to be supplied by the requestor of the service. The value is that of the Invoke-ID parameter of the RO-INVOKE indication primitive of the parent-operation.

10.1.1.6 *Priority*

This parameter defines the priority assigned to the transfer of the corresponding APDU with respect to the other APDUs to be exchanged between the AEs. The lower the value, the higher the priority. If several APDUs with the same priority are awaiting transfer, they are transferred "first in, first out".

Note 1 – The Priority parameter has an effect in the case of a two-way alternate association in that it prioritizes the sending of APDUs, and may be used to determine when to request the Turn to send APDUs. The Priority parameter may also have a local effect in the case of a two-way simultaneous association.

Note 2 – The Priority of a reply (RO-RESULT, RO-ERROR, and RO-REJECT-U) should normally be higher (lower in value) than the priority of the corresponding invocation.

10.2 *RO-RESULT service*

The RO-RESULT service is used by a ROSE-user to reply to a previous RO-INVOKE indication in the case of a successfully performed operation. This service is a non-confirmed service.

The related service structure consists of two service-primitives, as illustrated in Figure 6/X.219.

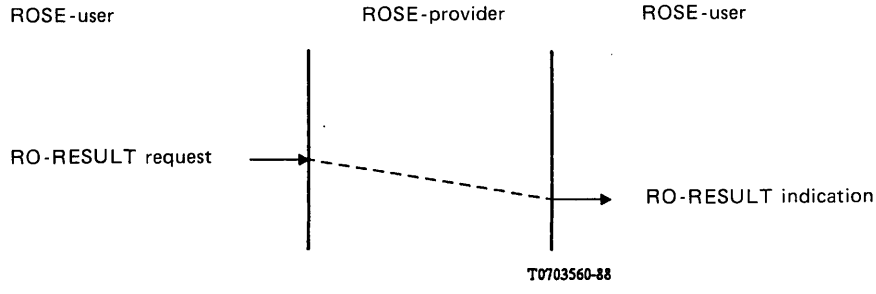


FIGURE 6/X.219

RO-RESULT Service-primitives

10.2.1 *RO-RESULT Parameters*

Table 3/X.219 lists the RO-RESULT service parameters.

TABLE 3/X.219

RO-RESULT Parameters

Parameter name	Request	Indication
Operation-value	U	C(=)
Result	U	C(=)
Invoke-ID	M	M(=)
Priority	U	

10.2.1.1 *Operation-value*

This parameter is the identifier of an invoked and successfully performed operation. This parameter has to be supplied by the requestor of the service. The value is that of the corresponding RO-INVOKE indication primitive. This parameter shall be present only if the Result parameter is present.

10.2.1.2 *Result*

This parameter is the result of an invoked and successfully performed operation. The type has to be agreed between the ROSE users. This parameter has to be supplied by the requestor of the service.

10.2.1.3 *Invoke-ID*

This parameter identifies the corresponding invocation (see § 10.1.1.4). This parameter has to be supplied by the requestor of the service. The value is that of the corresponding RO-INVOKE indication primitive.

10.2.1.4 *Priority*

This parameter defines the priority assigned to the transfer of the corresponding APDU (see § 10.1.1.6).

10.3 *RO-ERROR service*

The RO-ERROR service is used by a ROSE-user to reply to a previous RO-INVOKE indication in the case of an unsuccessfully performed operation. This service is a non-confirmed service.

The related service structure consists of two service-primitives as illustrated in Figure 7/X.219.

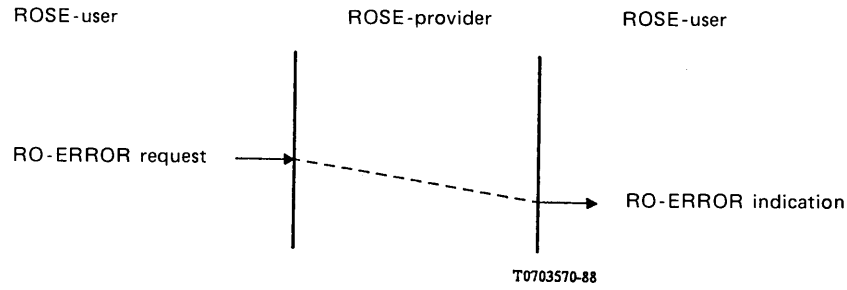


FIGURE 7/X.219

RO-ERROR Service-primitives

10.3.1 *RO-ERROR Parameters*

Table 4/X.219 lists the RO-ERROR service parameters.

TABLE 4/X.219

RO-ERROR Parameters

Parameter name	Request	Indication
Error-value	M	M(=)
Error-parameter	U	C(=)
Invoke-ID	M	M(=)
Priority	U	

10.3.1.1 *Error-value*

This parameter identifies the error that occurred during the execution of the operation. The value has to be agreed between the ROSE-users. This parameter has to be supplied by the requestor of the service.

10.3.1.2 *Error-parameter*

This parameter provides additional information about the error. The type (if any) has to be agreed between the ROSE-users. This parameter has to be supplied by the requestor of the service.

10.3.1.3 *Invoke-ID*

This parameter identifies the corresponding invocation (see § 10.1.1.4). This parameter has to be supplied by the requestor of the service. The value is that of the corresponding RO-INVOKE indication primitive.

10.3.1.4 *Priority*

This parameter defines the priority assigned to the transfer of the corresponding APDU (see § 10.1.1.6).

10.4 *RO-REJECT-U*

The RO-REJECT-U service is used by a ROSE-user to reject a request (RO-INVOKE indication) of the other ROSE-user if it has detected a problem. The RO-REJECT-U service may also be used by a ROSE-user to reject a reply (RO-RESULT indication, RO-ERROR indication) from the other ROSE-user. However, to avoid violating the sequencing rules of other ASEs in some application contexts, a ROSE-user may choose not to use the RO-REJECT-U service to reject replies. This service is a non-confirmed service.

The related service structure consists of two service-primitives, as illustrated in Figure 8/X.219.

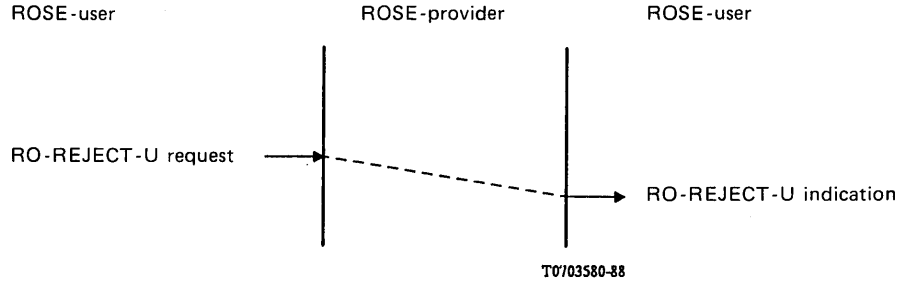


FIGURE 8/X.219

RO-REJECT-U Service-primitives

10.4.1 *RO-REJECT-U Parameters*

Table 5/X.219 lists the RO-REJECT-U service parameters.

TABLE 5/X.219

RO-REJECT-U Parameters

Parameter name	Request	Indication
Reject-reason	M	M(=)
Invoke-ID	M	M(=)
Priority	U	

10.4.1.1 *Reject-reason*

This parameter specifies the reason for rejection as follows:

a) *Invoke-problem:* user-reject of an RO-INVOKE indication primitive with values:
   - duplication-invocation
     signifies that the Invoke-ID parameter violates the assignment rules of § 10.1.1.4.
   - 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 Linked-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-user.

This parameter has to be supplied by the requestor of the service.

10.4.1.2 *Invoke-ID*

This parameter identifies the corresponding invocation (see § 10.1.1.4). This parameter has to be supplied by the requestor of the service. The value is that of the rejected RO-INVOKE indication, RO-RESULT indication, or RO-ERROR indication primitive.

10.4.1.3 *Priority*

This parameter defines the priority assigned to the transfer of the corresponding APDU (see § 10.1.1.6).

10.5 *RO-REJECT-P*

The RO-REJECT-P service is used to advise a ROSE-user of a problem detected by the ROSE-provider. This service is a provider-initiated service

The related service structure consists of a single service-primitive, as illustrated in Figure 9/X.219.

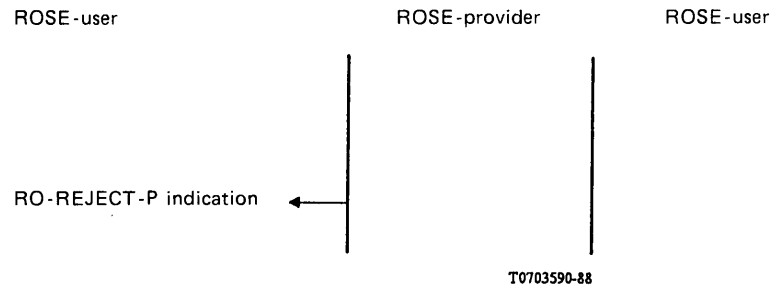


FIGURE 9/X.219

RO-REJECT-P Service-primitive

10.5.1 *RO-REJECT-P Parameters*

Table 6/X.219 lists the RO-REJECT-P service parameters.

TABLE 6/X.219

RO-REJECT-P Parameters

Parameter name	Indication
Invoke-ID	O
Returned-parameters	O
Reject-reason	O

10.5.1.1 *Invoke-ID*

This parameter identifies the corresponding invocation (see § 10.1.1.4). This parameter is supplied by the ROSE-provider. The value is that of the rejected RO-INVOKE request, RO-RESULT request, RO-ERROR request or RO-REJECT-U request primitives. This parameter may be omitted if an Invoke-ID is not available.

10.5.1.2 *Returned-parameters*

This parameter contains the parameters of the RO-INVOKE request, RO-RESULT request, RO-ERROR request or RO-REJECT-U request primitives, if the corresponding APDU could not be transferred by the ROSE-provider. This parameter and the parameter Reject-reason are mutually exclusive.

10.5.1.3 *Reject-reason*

This parameter specifies the reason for rejection as follows:

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 Recommendation X.229
- mistyped-APDU:
  signifies that the structure of the APDU does not conform to Recommendation X.229.
- badly-structured-APDU:
  signifies that the structure of the APDU does not conform to the standard notation and encoding, defined in Recommendation X.208 and X.209.

This parameter is supplied by the ROSE-provider. This parameter and the parameter Returned-parameters are mutually exclusive.

11 Mapping of Notation on Service

11.1 *Application Context and Operations*

This section describes how an application-context is specified by means of the notation provided by the macros defined in § 9.

Such an application-context specification consists of:

a) a bind-operation specified by means of the BIND macro, and

b) an unbind-operation specified by means of the UNBIND macro, and

c) a set of operations specified by means of the OPERATION macro, and

d) a set of errors related to operations and specified by means of the ERROR macro.

The Remote Operations (i.e. bind-operation, unbind-operation and operations) may be invoked by the user-element.

An association-initiating user-element established an application-association by invoking a bind-operation. If the application-association is established, operations may be invoked by the user-element. When the association-initiating user-element wishes to release the application-association, it invokes an unbind-operation.

11.2 *Mapping of Remote Operations on ACSE Services, RTSE Services, and ROSE Services*

The bind-operation and the unbind-operation are mapped either on ACSE services, or the RTSE services.

The operations are mapped on the ROSE services.

11.2.1 *Mapping on ACSE services*

The bind-operation is mapped on the A-ASSOCIATE services and the unbind-operation mapped on the A-RELEASE service.

11.2.1.1 *Mapping of a Bind-operation*

A bind-operation is mapped on the A-ASSOCIATE service.

11.2.1.1.1. *Invocation of a Bind-operation*

The invocation of a bind-operation is mapped on the A-ASSOCIATE request and A-ASSOCIATE indication service primitives.

The argument value of the bind-operation is mapped on the User Information parameter of the service primitives.

11.2.1.1.2 *Reply of a Bind-operation*

The reply of a bind-operation is mapped on the A-ASSOCIATE response and A-ASSOCIATE confirm service primitives.

If the bind-operation was successfully performed, the Result parameter of the service primitives is "accepted", and the result value of the bind-operation is mapped on the User Information parameter of the service primitives.

If the bind-operation was not successfully performed, the Result parameter value of the service primitives is "rejected (permanent)", and the error value of the bind-operation is mapped on the User Information parameter of the service primitives.

11.2.1.2 *Mapping of a Unbind-operation*

An unbind-operation is mapped on the A-RELEASE service.

11.2.1.2.1 *Invocation of an Unbind-operation*

The invocation of an unbind-operation is mapped on the A-RELEASE request and the A-RELEASE indication service primitives.

The argument value of the unbind-operation is mapped on the User Information parameter of the service primitives. The Reason parameter value of the service primitives is "normal".

11.2.1.2.2 *Reply of an Unbind-operation*

The reply of an unbind-operation is mapped on the A-RELEASE response and A-RELEASE confirm service primitives.

If the unbind-operation was successfully performed, the Reason parameter value of the service primitives is "normal", the result value of the unbind-operation is mapped on the User Information parameter of the service primitives, and the Result parameter of the service primitives is "affirmative".

If the unbind-operation was not successfully performed, the Reason parameter value of the service primitives is "not finished", the error value of the unbind-operation is mapped on the User Information parameter of the service primitives, and the Result parameter of the service primitives is "affirmative".

11.2.2 *Mapping on RTSE services*

The bind-operation is mapped on the RT-OPEN service and the unbind-operation mapped on the RT-CLOSE service.

11.2.2.1 *Mapping of a Bind-operation*

A bind-operation is mapped on the RT-OPEN service.

11.2.2.1.1 *Invocation of a Bind-operation*

The invocation of a bind-operation is mapped on the RT-OPEN request and RT-OPEN indication service primitives.

The argument value of the bind-operation is mapped on the User-data parameter of the service primitives. The Dialogue-mode parameter value is "two-way-alternate".

11.2.2.1.2 *Reply of a Bind-operation*

The reply of a bind-operation is mapped on the RT-OPEN response and RT-OPEN confirm service primitives.

If the bind-operation was successfully performed, the Result parameter of the service primitives is "accepted", and the result value of the bind-operation is mapped on the User-data parameter of the service primitives.

If the bind-operation was not successfully performed, the Result parameter value of the service primitives is "rejected (permanent)", and the error value of the bind-operation is mapped on the User data parameter of the service primitives.

11.2.2.2 *Mapping of an Unbind-operation*

An unbind-operation is mapped on the RT-CLOSE service.

11.2.2.2.1 *Invocation of an Unbind-operation*

The invocation of an unbind-operation is mapped on the RT-CLOSE request and the RT-CLOSE indication service primitives.

The argument value of the unbind-operation is mapped on the User-data parameter of the service primitives. The Reason parameter value of the service primitives is "normal".

11.2.2.2.2 *Reply of an Unbind-operation*

The reply of an unbind-operation is mapped on the RT-CLOSE response and RT-CLOSE confirm service primitives.

If the unbind-operation was successfully performed, the Reason parameter value of the service primitives is "normal", and the result value of the unbind-operation is mapped on the User-data parameter of the service primitives.

If the unbind-operation was not successfully performed, the Reason parameter value of the service primitives is "not finished", and the error value of the unbind-operation is mapped on the User-data parameter of the service primitives.

11.2.3 *Mapping on ROSE services*

An operation is mapped on the ROSE services.

11.2.3.1 *Invocation of an Operation*

The invocation of an operation is mapped on the RO-INVOKE service.

The value assigned to the operation is mapped on the Operation-value parameter of that service. The value of the Named-Type in the ARGUMENT clause of the OPERATION macro is mapped on the Argument parameter of that service.

11.2.3.2 *Reply of an operation*

If an operation was succesfully performed, the reply is mapped on the RO-RESULT service.

The value of the Named-Type in the RESULT clause of the OPERATION macro is mapped on the Result parameter of that service.

If an operation was not successfully performed, the reply is mapped on the RO-ERROR service.

In this case one of the errors in the Identifier List of Error Names in the ERROR clause of the OPERATION macro may be applied. The value assigned to the applied error is mapped on the Error parameter of that service. The value of the Named-Type in the PARAMETER clause of the ERROR macro of the applied error is mapped on the Error-parameter parameter of that service.

12 Sequencing Information

This section defines the interaction among the Remote Operations, and the interaction among the ACSE service and the ROSE services.

12.1 *Sequencing Information for Remote Operations*

12.1.1 *Bind-operation*

12.1.1.1 *Usage Restrictions*

A bind-operation is not used on an established application-association. A successfully performed bind-operation establishes an application-association.

12.1.1.2 *Disrupted Remote Operation*

The bind-operation does not disrupt any Remote Operation.

12.1.1.3 *Disrupting Remote Operations*

There are no disrupting Remote Operations.

12.1.1.4 *Collisions*

A bind-operation collision results when the user-elements in both AEs simultaneously invoke a bind-operation on each other. In this case two independent application-associations are established.

12.1.2 *Unbind-operation*

12.1.2.1 *Usage Restrictions*

An unbind-operation is only used on an established application-association. It is only used by the user-element which invoked the bind-operation. It is only used when no replies from Operation Class 1 or 2 operations are outstanding.

The application-association ceases to be established no matter whether the unbind-operation is performed successfully or not.

12.1.2.2 *Disrupted Remote Operations*

An unbind-operation does not disrupt Remote Operations in the case of Assoication Class 1 and Operation Class 1 or 2 operations.

In all other cases an unbind-operation may disrupt operations. However, if in a specific application-context Operation Classes 3, 4 or 5 and/or Association Class 2 or 3 are used, it is assumed that either the disruption is acceptable or the application-context provides operations to avoid the disruption.

12.1.2.3 *Disrupting Remote Operations*

There are no disrupting Remote Operations.

12.1.2.4 *Collisions*

Because only the association-initiator may release the application-association, there is no collision.

12.1.3 *Operations*

12.1.3.1 *Usage Restrictions*

Operations are only used on an established application-association.

12.1.3.2 *Disrupted Remote Operations*

Operations do not disrupt any Remote Operations.

12.1.3.3 *Disrupting Remote Operations*

Operations may be disrupted by an unbind-operation (see § 12.1.2.2).

12.1.3.4 *Collisions*

There are no collisions of operations.

12.1.4 *Further Sequencing Information*

Disrupting services are not visible at the operation-interface. However operations may be disrupted by services (see § 12.2).

Bind-operations and unbind-operations are disrupted by the A-ABORT, A-P-ABORT, RT-U-ABORT and RT-P-ABORT services.

Operations are disrupted by the A-ABORT, A-P-ABORT, RT-U-ABORT, RT-P-ABORT, RO-REJECT-U and RO-REJECT-P services.

In addition an operation may be disrupted by the A-RELEASE service. But this reflects only the disruption of an operation by an unbind-operation. The disrupted operations are not considered in the disrupted services of § 12.2.

Because all Remote Operations are mapped on services, and disrupting Remote Operations (unbind-operation) are represented by services, no disrupting Remote Operations are considered in the disrupting services clauses of § 12.2.

12.2 *Sequencing Informations for Services*

12.2.1 *ACSE Services*

The sequencing information for ACSE services are described in Recommendation X.217. Additional information is provided in this clause.

12.2.1.1 *Disrupted ROSE Services*

In addition to the disrupted services defined in Recommendation X.217 all ROSE services except the RO-REJECT-P service are disrupted by the A-ABORT and A-P-ABORT services and may be disrupted by the A-RELEASE service (see § 12.2.3.6).

12.2.1.2 *Disrupting ROSE Services*

There are no disrupting ROSE services.

12.2.2 *RTSE Services*

The sequencing information for RTSE services are described in Recommendation X.218. Additional information is provided in this section.

12.2.2.1 *Disrupted ROSE Services*

In addition to the disrupted services defined in Recommendation X.218 all ROSE services except the RO-REJECT-P service are disrupted by the RT-U-ABORT, RT-P-ABORT and the negative RT-TRANSFER confirm services.

12.2.2.2 *Disrupting ROSE Services*

There are no disrupting ROSE services.

12.2.3 *ROSE Services*

This section describes the interaction among the ROSE services. Interactions with ACSE services are described in § 12.2.1, and interactions with RTSE services are described in § 12.2.2.

12.2.3.1 *RO-INVOKE Service*

12.2.3.1.1 *Type of Service*

The RO-INVOKE service is a non-confirmed service.

12.2.3.1.2 *Usage Restriction*

The RO-INVOKE service is only used on an established application-association.

12.2.3.1.3 *Disrupted Services*

The RO-INVOKE service does not disrupt any services.

12.2.3.1.4 *Disrupting Services*

The RO-INVOKE service is disrupted by the RO-REJECT-P service.

12.2.3.1.5 *Collision*

There are no collisions of the RO-INVOKE service.

12.2.3.2 *RO-RESULT*

12.2.3.2.1 *Type of Service*

The RO-RESULT service is a non-confirmed service.

12.2.3.2.2 *Usage Restriction*

The RO-RESULT service is only used on an established application-association and in reply to an RO-INVOKE service.

12.2.3.2.3 *Disrupted Services*

The RO-RESULT services does not disrupt any services.

12.2.3.2.4 *Disrupting Services*

The RO-RESULT service is disrupted by the RO-REJECT-P service.

12.2.3.2.5 *Collisions*

There are no collisions of the RO-RESULT service.

12.2.3.3 *RO-ERROR*

12.2.3.3.1 *Type of Service*

The RO-ERROR service is a non-confirmed service.

12.2.3.3.2 *Usage Restriction*

The RO-ERROR service is only used on an established application-association and in reply to an RO-INVOKE service.

12.2.3.3.3 *Disrupted services*

The RO-ERROR service does not disrupt any service.

12.2.3.3.4 *Disrupting Services*

The RO-ERROR service is disrupted by the RO-REJECT-P service.

12.2.3.3.5 *Collisions*

There are no collisions of the RO-ERROR service.

12.2.3.4 *RO-REJECT-U*

12.2.3.4.1 *Type of Service*

The RO-REJECT-U service is a non-confirmed service.

12.2.3.4.2 *Usage Restriction*

The RO-REJECT-U service is only used on an established application-association and in reply to RO-INVOKE, RO-RESULT and RO-ERROR services.

12.2.3.4.3 *Disrupted Services*

The RO-REJECT-U service does not disrupt any service.

12.2.3.4.4 *Disrupting Services*

The RO-REJECT-U service is disrupted by the RO-REJECT-P service.

12.2.3.4.5 *Collisions*

There are no collisions of the RO-REJECT-U service.

12.2.3.5 *RO-REJECT-P*

12.2.3.5.1 *Type of Service*

The RO-REJECT-P service is a provider initiated service.

12.2.3.5.2 *Usage Restriction*

Not applicable.

12.2.3.5.3 *Disrupted Services*

The RO-REJECT-P service disrupts all other ROSE services.

12.2.3.5.4 *Disrupting Services*

The RO-REJECT-P service is not disrupted by any other service.

12.2.3.5.5 *Collision*

If the ACSE service or an RTSE service cause an abort or the release of an application-association, it is a local matter to inform the service-user about outstanding RO-REJECT-P services for Returned-parameters.

12.2.3.6 *Additional Sequencing Information*

The usage restrictions for unbind-operations (§ 12.1.2.1) and the mapping of unbind-operations onto the A-RELEASE service prevents the disruption of ROSE services, if only Association Class 1 and Operation Classes 1 and 2 are used.

If Association Classes 2 or 3, or Operation Class 3, 4 or 5 are used, the A-RELEASE service may disrupt ROSE services. In this case it is the responsibility of the application-context designer, either to accept this disruption or to provide means (e.g. operations) to prepare for the release of an application-association.

ANNEX A

(to Recommendation X.219)

Notation Supporting the Specification of Application-service-elements and Application Contexts

This Annex is an integral part of this Recommendation.

This Annex provides a notation supporting the specification of application-service-elements and application contexts which are specified by means of the RO-notation. The RO-notation may be used to specify the bind-operation and the unbind-operation of an application context. Additionally the RQ-notation may be used to specify operation types and error types of several application-service-elements (ROSE-user ASEs). If the RQ-notation is combined with other notations, other specification tools defined elsewhere might be used.

This Annex defines two macros supporting the specification of application-service-elements and application contexts. The formal definition of these macros is shown in Figure A-1/X.219.

A.1 *Application-service-elements*

The notation supports the unique identification of an ASE.

If an ASE is an ROSE-user, the notation additionally supports the specification of the characteristics of the ASE. The operation-interface and the protocol of an ROSE-user ASE are specified by a set of operation types and a set of error types.

The protocol specified for a specific ASE may be inherently:

a) symmetric, or

b) asymmetric.

In the symmetric case both ASE-users may invoke the same set of operation types.

In the asymmetric case one ASE-user (called supplier in the context of this Annex) provides some information-processing functionality which is used by the peer ASE-user (called consumer in the context of this Annex). In this case a specific operation type may be:

1) invoked by the consumer, and/or

2) invoked by the supplier

of the information-processing functionality.

Note – A particular allocation of the terms "supplier" and "consumer" is often intuitive. One might naturally consider a file system, e.g., to be called a supplier and its user to be called a consumer. Strictly speaking, however, the asignment of the two terms is arbitrary.

If an ASE uses the concept of linked-operations, a specific operation type may be invoked as a child-operation. The invoker of the child-operation is the performer of the linked parent-operation. Operation types solely invoked as child-operations shall not be included in the ASE specification, they are listed in the type specification of their parent-operations.

The error types the operations may report are not included in the ASE specification, they are listed in the type specification of the operations.

The set of the remaining operations (operations not solely invoked as child-operations), and who is allowed to invoke this operations may be reflected by a formal notation specifying the ROSE-user ASE.

A.1.1 *Specification of an Application-service-element*

An ASE may be specified by a formal notation supported by the **APPLICATION-SERVICE-ELEMENT** macro (see Figure A-1/X.219).

```
Remote-Operations-Notation-extension {joint-iso-ccitt remote-operations(4) notation-extension(2)}

DEFINITIONS ::=

BEGIN


EXPORTS  APPLICATION-SERVICE-ELEMENT, APPLICATION-CONTEXT, aCSE;


IMPORTS  OPERATION, BIND, UNBIND FROM  Remote-Operation-Notation
                                       {joint-iso-ccitt remote-operations(4) notation(0)};

-- macro definition for ASEs


APPLICATION-SERVICE-ELEMENT-MACRO ::=

BEGIN


TYPE NOTATION              ::=  SymmetricAse | ConsumerInvokes SupplierInvokes | empty

VALUE NOTATION             ::=  value (VALUE OBJECT IDENTIFIER)

SymmetricAse               ::=  "OPERATIONS" "{"OperationList"}"

ConsumerInvokes            ::=  "CONSUMER INVOKES" "{"OperationList"}" | empty

SupplierInvokes            ::=  "SUPPLIER INVOKES" "{"OperationList"}" | empty

OperationList              ::=  Operation | OperationList "," Operation

Operation                  ::=  value (OPERATION)


END

aCSE APPLICATION-SERVICE-ELEMENT ::= {joint-iso-ccitt remote-operations(4) aseID-ACSE(4)}

-- Remote Operations Notation extension continued
```

FIGURE A-1/X.219 (Part 1 of 2)

Formal Definition of ASE and Application-context Data Types

The type notation of the **APPLICATION-SERVICE-ELEMENT** macro enables the specification of an ASE. The value notation of the **APPLICATION-SERVICE-ELEMENT** macro enables the specification of a unique identifier for the ASE.

The notation for an ASE type is the keyword **APPLICATION-SERVICE-ELEMENT** optionally followed by the specification of operations.

If the protocol specified for the ASE is symmetric, the keyword **OPERATIONS** and the reference names of the operations are specified.

If the protocol specified for the ASE is asymmetric, the keywords **CONSUMER INVOKES** and the reference names of the operations the consumer may invoke, and/or the keywords **SUPPLIER INVOKES** and the reference names of the operations the supplier may invoke, are specified.

A.2 *Application Contexts*

In the context of this Annex an application context explicitly identifies:

a) a bind-operation,

b) an unbind-operation, and

c) a set of application-service-elements.

and requires one or more identified abstract-syntaxes.

The set of application-service-elements contains:

1) ASEs not using the RO-notation, i.e. the ACSE, optionally the RTSE, and optionally others; and

2) optionally the ROSE and ROSE-user ASEs.

If the application context contains ROSE-user ASEs with an asymmetrical protocol, the notation supports the specification whether the association-initiator or the association-responder is the consumer of such an ASE.

The identification of the bind-operation, the unbind-operation, the application-service-elements, and the abstract-syntaxes may be reflected by a formal notation specifying the application context.

A.2.1 *Specification of an Application Context*

An application context may be specified by a formal notation supported by the **APPLICATION-CONTEXT** macro (see Figure A-1/X.219).

The type notation of the **APPLICATION-CONTEXT** macro enables the specification of the application context. The value notation of the **APPLICATION-CONTEXT** macro enables the specification of a unique identifier for the application context.

The notation for an application context type is the keyword **APPLICATION-CONTEXT**, followed by the keywords **APPLICATION SERVICE ELEMENTS** and the reference names of ASEs not using the RO-notation, followed by the keyword **BIND** and the reference name of the bind-operation type, followed by the keyword **UNBIND** and the reference name of the unbind-operation type, optionally followed by the specification of ASEs using operations, followed by the keywords **ABSTRACT SYNTAXES** and the reference names of the abstract syntaxes.

If the application context contains ROSE-user ASEs, the keywords **REMOTE OPERATIONS** and the reference name of the ROSE, followed by the specification of ASEs with a symmetrical protocol, and/or the specification of ASEs with an asymmetrical protocol is used. The specification of ASEs with a symmetrical protocol is the keywords **OPERATIONS OF** and the reference names of that ASEs. The specification of ASEs with an asymmetrical protocol is the keywords **INITIATOR CONSUMER OF** and the reference names of ASEs the consumer of which is the association-initiator, and/or the keywords **RESPONDER CONSUMER OF** and the reference names of ASEs the consumer of which is the association-responder.

A.2.2 *Mapping of Notation onto Service*

The application context identifier and the list of abstract syntax names specified by means of the **APPLICATION-CONTEXT** macro are mapped either on the RT-OPEN services of RTSE, if RTSE is included in the application context; or else on the A-ASSOCIATE services of ACSE.

The application context value is mapped onto the Application Context Name parameter of the RT-OPEN or A-ASSOCIATE services.

The abstract syntax names are mapped onto the Presentation Context Definition List parameter and the Presentation Context Definition Result List of the RT-OPEN or A-ASSOCIATE services.

```
-- Remote Operations Notation extension continued

-- macro definition for application contexts

APPLICATION-CONTEXT MACRO ::=
BEGIN

TYPE NOTATION          ::=  NonROelements Binding ROelements AbstractSyntaxes
VALUE NOTATION         ::=  value (VALUE OBJECT IDENTIFIER)
NonROelements          ::=  "APPLICATION SERVICE ELEMENTS" "{"AseList"}"
Binding                ::=  "BIND"   type -- shall reference a bind-operation type
                            "UNBIND" type -- shall reference an unbind-operation type
ROelements             ::=  "REMOTE OPERATIONS" "{"AseID"}" -- identifying ROSE
                            SymmetricAses AsymmetricAses | empty
SymmetricAses          ::=  "OPERATIONS OF" "{"AseList"}" | empty
AsymmetricAses         ::=  InitiatorConsumerOf ResponderConsumerOf
InitiatorConsumerOf    ::=  "INITIATOR CONSUMER OF" "{"AseList"}" | empty
ResponderConsumerOf    ::=  "RESPONDER CONSUMER OF" "{"AseList"}" | empty
AbstractSyntaxes       ::=  "ABSTRACT SYNTAXES" "{"AbstractSyntaxList"}"
AseList                ::=  AseID | AseList "," AseID
AseID                  ::=  value (APPLICATION-SERVICE-ELEMENT)
AbstractSyntaxtList    ::=  AbstractSyntax | AbstractSyntaxList "," AbstractSyntax
AbstractSyntax         ::=  value (OBJECT IDENTIFIER) -- identifying abstract syntax

END

END                    --  end of Remote Operations Notation extension
```

FIGURE A-1/X.219 (Part 2 of 2)

Formal Definition of ASE and Application-context Data Types

ANNEX B

(to Recommendation X.219)

Guidelines for Application Protocol Designers on the Use of ROSE

This Annex is not part of this Recommendation.

This Annex provides examples and guidelines for application protocol designers on the user of ROSE.

B.1 *Examples for Operations and Errors*

This section provides examples for the definition of operations and errors.

B.1.1 *Operation Classes*

This clause provides examples for the definition of operations of Operation Class 1 to 5.

The operation **operationExample12** of Operation Class 1 or 2 (see example below) reports success (result of type **ArgumentType12**) or failure (errors **errorExample1** or **errorExample2**). The argument of the operation **operationExample12** is of type **ArgumentType12**. The value of the operation **operationExample12** is **1**.

```
operationExample12    OPERATION
                      ARGUMENT        ArgumentType12
                      RESULT          ResultType12
                      ERROR           {errorExample1, errorExample2}
                      ::= 1
```

The operation **operationExample3** of Operation Class 3 (see example below) reports failure (error **errorExample1**) only, if any. The argument of the operation **operationExample3** is of type **ArgumentType3**. The value of the operation **operationExample3** is **2**.

```
operationExample3     OPERATION
                      ARGUMENT        ArgumentType3
                      ERROR           {errorExample1}
                      ::= 2
```

The **operationExample4** operation of Operation Class 4 (see example below) reports success (result of type **ResultType4**) only. The argument of the operation **operationExample4** is of type **ArgumentType4**. The value of the operation **operationExample4** is **3**.

```
operationExample4     OPERATION
                      ARGUMENT        ArgumentType4
                      RESULT          ResultType4
                      ::= 3
```

The operations **operationExample51** and **operationExample52** of Operation Class 5 (see example below) reports no outcome. The argument of the operation **operationExample51** is of type **ArgumentType4**, the operation **operationExample52** has no argument. The value of the operation **operationExample51** is **4**, the value of the operation **operationExample52** is **5**.

```
operationExample51    OPERATION
                      ARGUMENT        ArgumentType4
                      ::= 4

operationExample52    OPERATION
                      ::= 5
```

B.1.2 *Linked-operations*

The example below shows the definition of a set of linked-operations consisting of the parent-operation **parent-op12** and the child-operations **operationExample51** and **operationExample52**.

```
parent-op12           OPERATION
                      ARGUMENT        ArgumentType12
                      RESULT          ResultType12
                      ERROR           {errorExample1, errorExample2}
                      LINKED          {operationExample51, operationExample52}
                      ::= 6
```

B.1.3 *Errors*

Errors (see **errorExample1** and **errorExample2** below) reports failure. The parameter of the error **errorExample1** is of type **ParameterType1**, the error **errorExample2** has no parameter. The value of the error **errorExample1** is **1**, the value of the error **errorExample2** is **2**.

```
errorExample1         ERROR
                      PARAMETER       ParameterType1
                      ::= 1

errorExample2         ERROR
                      ::= 2
```

B.2 *Examples for Bind-operations and Unbind-operations*

This clause provides examples for the definition of bind-operations and unbind-operations.

B.2.1 *Bind-operations*

Bind-operations are used to establish an application-association.

The request of the bind-operation **BindExample1** to establish an application-association is accompanied by the argument of type **BindArgumentType1**. The positive response to the application-association establishment is accompanied by the result of type **BindResultType1**. The negative response to the application-association establishment is accompanied by the bind-error of type **BindErrorType1**.

```
BindExample1 ::=    BIND
                    ARGUMENT        BindArgumentType1
                    RESULT          BindResultType1
                    BIND-ERROR      BindErrorType1
```

The request of the bind-operation **BindExample2** to establish an application-association is accompanied by the argument of type **BindArgumentType1**. The positive response to the application-association establishment is not accompanied by any user data. The negative response to the application-association establishment is accompanied by the bind-error of type **BindErrorType1**.

```
BindExample2 ::=    BIND
                    ARGUMENT        BindArgumentType1
                    BIND-ERROR      BindErrorType1
```

Note that argument, result and bind-error of a bind-operation are optional. Neither the request of the bind-operation **BindExample3** to establish an application-association nor the response to the application-association establishment are accompanied by any user data.

```
BindExample3 ::=    BIND
```

B.2.2 *Unbind-operations*

Unbind-operations are used to release an application-association.

The request of the unbind-operation **UnbindExample1** to release an application-association is accompanied by the argument of type **UnbindArgumentType1**. The response to the application-association release is accompanied either by the result of type **UnbindResultType1** or by the unbind-error of type **UnbindErrorType1**.

```
UnbindExample1 ::=  UNBIND
                    ARGUMENT        UnbindArgumentType1
                    RESULT          UnbindResultType1
                    UNBIND-ERROR    UnbindErrorType1
```

The request of the unbind-operation **UnbindExample2** to release an application-association is accompanied by the argument of type **UnbindArgumentType 1**. The response to the application-association release is optionally accompanied by the unbind-error of type **UnbindErrorType1**.

```
UnbindExample2 ::=  UNBIND
                    ARGUMENT        UnbindArgumentType1
                    UNBIND-ERROR    UnbindErrorType1
```

Note that argument, result and unbind-error of an unbind-operation are optional. Neither the request of the bind-operation **UnbindExample3** to release an application-association nor the response to the application-association release are accompanied by any user data.

```
UnbindExample3 ::=  UNBIND
```

B.3 *Export and Import of Operations and Errors*

This clause gives examples how to export and import operations and errors.

The example below shows how to export operation and errors. The **operation10** and **error10** have local values. **OperationTypeA** and **ErrorTypeA** are types, and specific values have to be assigned in the importing modules. The **operation11** and **error11** have globally unique values.

```
ExportingModule {objectidentifier1} DEFINITIONS ::=
BEGIN

        EXPORTS         operation10, OperationTypeA, operation11, error10, ErrorTypeA, error11;
        IMPORTS         OPERATION, ERROR, BIND, UNBIND
                        FROM Remote-Operation-Notation
                                {joint-iso-ccitt remote-operation(4) notation(0)};

        operation10     OPERATION
                        ARGUMENT        ArgumentType10
                        RESULT          ResultType10
                        ERROR           {error10}
                        ::= 10

        OperationTypeA  ::=
                        OPERATION
                        ARGUMENT        ArgumentTypeA
                        RESULT          ResultTypeA

        operation11     OPERATION
                        ARGUMENT        ArgumentType11
                        RESULT          ResultType11
                        ERROR           {error11}
                        ::= {objectidentifier2component11}

        error10         ERROR
                        PARAMETER       ParameterType10
                        ::= 10

        errorTypeA      ::=
                        ERROR
                        PARAMETER       ParameterTypeA

        error11         ERROR
                        PARAMETER       ParameterType11
                        ::= {objectidentifier2component12}

END
```

The example below shows how to import operations and errors. The **operation10** and **erro10** have loval values defined in the exporting module. The designer of the importing module is responsible for the uniqueness of these values within the abstract syntax. The operation **operation13** is of type **operationTypeA** and has the value **13** assigned. The error **error13** is of type **ErrorTypeA** and has the value **13** assigned. The **operation11** and **error11** have globally unique values defined in the exporting module.

```
ImportingModule {objectidentifier3} DEFINITIONS ::=
BEGIN

        IMPORTS         operation10, OperationTypeA, operation11, error10, errorTypeA, error11
                        FROM ExportingModule {objectidentifier1};

        operation13     OperationTypeA  ::= 13
        error13         ErrorTypeA      ::= 13

END
```

B.4 *Definition of Application-service-elements*

This clause gives examples how to define application-service-elements. The examples refer to the operations and errors defined in the examples of clause B.1.

The application-service-element **element1** includes the operations **operationExample12** and **operationExample3** and the errors **errorExample1** and **errorExample2**. Note, the errors are included indirectly by the definition of the operations. The application-service-element **element1** is symmetric, i.e. both user-elements may invoke the operations **operationExample12** and **operationExample3**.

```
element1    APPLICATION-SERVICE-ELEMENT
            OPERATIONS {operationExample12, operationExample3}
            ::= {objectidentifierOfElement1}
```

The application-service-element **element2** includes the operations **operationExample3** and **operationExample4** and the error **errorExample1**. The application-service-element **element2** is asymmetric, i.e. only one user-element (that in the consumer role) may invoke the operations **operationExample3** and **operationExample4**.

element2 APPLICATION-SERVICE-ELEMENT
CONSUMER INVOKES {operationExample3, operationExample4}
::= {objectidentifierOfElement2}

The application-service-element **element3** includes the operations **operationExample12** and **operationExample51** and the errors **errorExample1** and **errorExample2**. The application-service-element **element3** is asymmetric, i.e. only one user-element (that in the supplier role) may invoke the operations **operationExample12** and **operationExample51**.

element3 APPLICATION-SERVICE-ELEMENT
SUPPLIER INVOKES {operationExample12, operationExample51}
::= {objectidentifierOfElement3}

The application-service-element **element4** includes the operations **parent-op12**, **operationExample51** and **operationExample52** and the errors **errorExample1** and **errorExample2**. Note, the child-operations **operationExample51** and **operationExample52** are included indirectly by the definition of the parent-operation **parent-op12**. The application-service-element **element4** is asymmetric, i.e. only one user-element (that in the consumer role) may invoke the operation **parent-op12** and only the other user-element (that in the supplier role) may invoke the child-operations **operationExample51** and **operationExample52** during the execution of the parent-operation **parent-op12**.

element4 APPLICATION-SERVICE-ELEMENT
CONSUMER INVOKES {parent-op12}
::= {objectidentifierOfElement4}

The application-service-element **element5** includes the same operations and the errors as the application-service-element **element4**. The only difference is, that the user-element in the supplier role may invoke the operation **operationExample52** either as a child-operation, or outside a set of linked-operations (as a non-child-operation).

element5 APPLICATION-SERVICE-ELEMENT
CONSUMER INVOKES {parent-op12}
SUPPLIER INVOKES {operationExample52}
::= {objectidentifierOfElement5}

B.5 *Definition of Application Contexts*

This clause gives examples how to define application contexts for several Association Classes. The examples refer to the application-service-element defined in the examples of clause B.4, and the bind-operations and unbind-operations defined in the examples of clause B.2.

The application context **context1** includes the application-service-elements ACSE, RTSE, ROSE, and **element2**; the bind-operation **BindExample1** and the unbind-operation **UnbindExample3**. The association-initiator may invoke the operations **operationExample3** and **operationExample4**. The association-responder is not allowed to invoke any operation (Association Class 1).

context1 APPLICATION-CONTEXT
APPLICATION SERVICE ELEMENTS {aCSE, rTSE}
BIND BindExample1
UNBIND UnbindExample3
REMOTE OPERATIONS {rOSE}
INITIATOR CONSUMER OF {element2}
ABSTRACT SYNTAXES { { joint-iso-ccitt association-control(2) abstractSyntax(1) apdus(0) version1(1)}, objectidentifierOfAbstracSyntax1}
::= {objectidentifierOfContext1}

The application context **context2** includes the application-service-elements ACSE, ROSE, **element2**, and **element3**; the bind-operation **BindExample1**; and the unbind-operation **UnbindExample3**. The association-responder may invoke the operations **operationExample3**, **operationExample4**, **operationExample12** and **operationExample51**. The association-initiator is not allowed to invoke any operation (Association Class 2).

```
context2    APPLICATION-CONTEXT
            APPLICATION SERVICE ELEMENTS    {aCSE}
            BIND                            BindExample1
            UNBIND                          UnbindExample3
            REMOTE OPERATIONS               {rOSE}
            INITIATOR CONSUMER OF           {element3}
            RESPONDER CONSUMER OF           {element2}
            ABSTRACT SYNTAXES               { { joint-iso-ccitt association-control(2) abstract-
                                            Syntax(1)
                                              apdus(0) version1(1)},
                                            objectidentifierOfAbstracSyntax2}
            ::= {objectidentifierOfContext2}
```

The application context **context3** includes the application-service-elements ACSE, RTSE, ROSE, and **element2**; the bind-operation **BindExample1**; and the unbind-operation **UnbindExample3**. The association-responder may invoke the operations **operationExample3** and **operationExample4**. The association-initiator is not allowed to invoke any operation (Association Class 2).

```
context3    APPLICATION-CONTEXT
            APPLICATION SERVICE ELEMENTS    {aCSE, rTSE}
            BIND                            BindExample1
            UNBIND                          UnbindExample3
            REMOTE OPERATIONS               {rOSE}
            RESPONDER CONSUMER OF           {element2}
            ABSTRACT SYNTAXES               {objectidentifierOfAbstracSyntax3}
            ::= {objectidentifierOfContext3}
```

The application context **context4** includes the application-service-elements ACSE, ROSE, and **element2**; the bind-operation **BindExample3**; and the unbind-operation **UnbindExample3**. The application context **context3** is symmetric, i.e. both the association-initiator and the association-responder may invoke the operations **operationExample12** and **operationExample3** (Association Class 3).

```
context4    APPLICATION-CONTEXT
            APPLICATION SERVICE ELEMENTS    {aCSE}
            BIND                            BindExample3
            UNBIND                          UnbindExample3
            REMOTE OPERATIONS               {rOSE}
            OPERATIONS OF                   {element1}
            ABSTRACT SYNTAXES               {objectidentifierOfAbstracSyntax4}
            ::= {objectidentifierOfContext4}
```

The last two examples assume a single named abstract syntax.

B.6 *Releasing Application-associations in an Orderly Way*

B.6.1 *Introduction*

This Recommendation defines five Operation Classes based upon the outcomes they report, and three Association Classes based upon whether the association-initiator, the association-responder, or both may invoke operations. This clause defines the rules that ensure orderly release of application-associations and the Operation Classes invoked over it.

B.6.2 *Objectives*

The rules of this paragraph are intended to achieve one of the following two objectives, depending upon the situation:

a) *The Exactly-Once Objective:* Ideally an application entity should be able to count on the invocation of an operation causing that operation to be performed exactly once, that is, not several times and not none at all.

b) *The At-Most-Once Objective:* In come circumstances the Exactly-Once Objective cannot be achieved. A lesser but still useful objective is that invoking an operation will cause that operation to be performed at most once, that is, perhaps not at all but never twice.

B.6.3 *Definition of Rules*

The following **general rules** apply in all circumstances:

G1 The performer shall report the result or error of each confirmed operation over the same application-association by means of which the operation was invoked.

G2 The initiator shall not release the application-association until all operations it invoked have been confirmed.

The following **specific rules** apply in certain circumstances:

S1 Each time it exercises the RO-INVOKE service, the invoker shall supply a different Invoke-ID (unless reattempting an invocation) even across a succession of application-associations. This enables the performer to achieve the At-Most-Once Objective by suppressing duplicates.

S2 If the performer encounters a duplicate Invoke-ID in the RO-INVOKE service, it shall exercise the RO-REJECT-U service with duplicate-invocation as the Reject-reason. This helps to achieve the Exactly-Once Objective.

S3 The association-initiator shall reject any invocations it has not performed before releasing the application-association.

S4 The association-initiator shall respond to any invocations it has performed before releasing the application-association.

B.6.4 *Application of Rules*

The general rules always apply. The specific rules govern application-associations of particular Association Classes, and operations (invoked by means of such associations) of particular Operation Classes, as follows:

a) *Application-associations of Association Class 1:* For operations of Operation Classes 1 and 2, no specific rules apply. For operations of Operation Class 3, 4 and 5, specific rules S1 and S2 apply.

b) *Application-associations of Association Class 2 and 3:* For operations of Operation Classes 1 and 2, specific rules S3 and S4 apply. (Any invocation issued by the association-responder after the association-initiator has issued a release are lost. Response-rejects may be lost as well.) For operations of Operation Class 3, 4, and 5, specific rules S1, S2, S3 and S4 apply.

For protocols containing only operations of Operation Classes 1 or 2, the only constraint upon the values of the Invoke-IDs the invoker supplies to the RO-INVOKE service is that they be distinct over the life of the application-association.

Application-entities make Invoke-IDs unique to an invoker and across consecutive application-associations by exchanging presentation-addresses at application-association establishment time and, for each presentation-address, making Invoke-IDs integers that increase monotonically over some reasonable long period of time.

To ensure that an operation of Operation Classes 3, 4, or 5 has been performed exactly once, an application-entity shall elicit the duplicate-invocation Reject-reason by invoking the operation twice (or more) with the same Invoke-ID. Otherwise, the At-Most-Once Objective is all that is assured, suggesting that, on average, the invoker does not care whether a non-confirmed operation is perfomed.

B.6.5 *Observations*

Every operation of Operation Classes 1 or 2 is performed exactly once.

The usefulness of non-confirmed operations or conditionally confirmed operations may depend on the specific application. Protocol designers should be advised to define only operations of Operation Classes 1 or 2 unless some very specialized requirements exists.

ISBN 92-61-03691-0