ECMA
EUROPEAN COMPUTER MANUFACTURERS ASSOCIATION

STANDARD ECMA-75

SESSION PROTOCOL

January 1982

Free copies of this document are available from ECMA,
European Computer Manufacturers Association
114 Rue du Rhône – 1204 Geneva (Switzerland)

ECMA

EUROPEAN COMPUTER MANUFACTURERS ASSOCIATION

STANDARD ECMA-75

SESSION PROTOCOL

January 1982

BRIEF HISTORY

Work on an ISO Reference Model for Interconnection started in 1977 with the formation of ISO/TC97/SC16 and the development of a Reference Model, now ISO DP 7498.

In ECMA, work on a Session Protocol to be designed in accordance with Layer 5 of the ISO Reference Model started in 1978 with the formation of SSP TG in TC23 (Open Systems Interconnection). The Session Protocol was designed to use and enhance the services provided by means of the Transport Protocol (Standard ECMA-72).

The principal characteristics of the services provided by the Session Protocol are:

- reliable and transparent data transfer,
- organized data transfer,
- synchronized data transfer.

Reliable and transparent data transfer is derived directly from the Transport Service. The other two characteristics are added value services provided by the Session Protocol.

This Standard ECMA-75 is based on the practical experience of ECMA member Companies world-wide, and on the result of their active participation in the current work of ISO, the CCITT and national standard bodies in Europe and USA.

This Standard ECMA-75 has been adopted by the General Assembly of ECMA on December 10, 1981.

TABLE OF CONTENTS

Table of Contents (cont'd)

1 General

1.1 INTRODUCTION

This Standard ECMA-75 is one of a set of Standard for Open Systems Interconnection. Open System Interconnection standards are intended to facilitate homogeneous interconnection between heterogeneous information processing systems. The standard is within the framework for the co-ordination of standards for Open Systems Interconnection which is defined by ISO/DP 7498.

This ECMA standard is based on the practical experience of ECMA member companies world-wide, and on the results of their active participation in the current work of ISO, the CCITT and national standards bodies in Europe and the USA. It represents a pragmatic and widely based consensus.

A particular emphasis of this standard is to specify the homogeneous externally visible and verifiable characteristics needed for interconnection compatibility, while avoiding unnecessary constraints upon and changes to the heterogeneous internal design and implementation of the information processing system to be interconnected.

In the interests of rapid and effective standardization, the standard is oriented towards urgent and well understood needs. It is intended to be capable of modular extension to cover future developments in technology and needs.

Section three of this standard (protocol) has precedence over section two (service) if any differences arise in interpreting their content.

1.2 SCOPE

This ECMA Session Protocol Standard:

- defines general session layer concepts (see section 1),
- defines abstract interactions between two session-service-users via the session service (see section 2),
- defines the protocol to support these services (see section 3),
- specifies the requirements for conformance with this protocol (see section 4).

This standard defines only what is needed for a basic session layer. It provides the consistent technical basis for further session layer standards with extended scope.

This standard defines what is needed for compatible interconnection between information processing systems. It does not define local interactions between a session-service-user and the session-service. It in no way defines interlayer interfaces.

This standard is for the session layer of Open Systems Interconnection (see ISO/DP 7498).

1.3 REFERENCES

ISO/DP 7498 Data Processing - Open Systems Interconnection - Basic Reference Model

ECMA-72 Transport Protocol

1.4 GENERAL OVERVIEW

The role of session protocol is to use and to enhance the services provided by means of transport protocol (see Standard ECMA-72).

The principal characteristics of the services provided by means of session protocol are:

- reliable and transparent data transfer,
- organized data transfer,
- synchronized data transfer.

Reliable and transparent data transfer is derived directly from the transport service. The other two characteristics are added value provided by session protocol.

The organization is mostly concerned with the orderly establishment and termination of session-connections, and with providing controls to assist structuring of the interactions communicated on the session-connection.

The synchronization is mostly concerned with effective ways of handling the uncertainties and inefficiencies caused by transit delay (defined in B.2.3).

1.5 CONCEPTS

1.5.1 Synchronization

A distinctive characteristic of the session-service is a systematic factoring out of complex transit delay effects (defined in B.2.3) relating to synchronization (defined in B.2.1) and performance optimization (see notes 1, 2 and 3).

NOTE 1

Transit delay causes uncertainty of synchronization and additional complexity in collision and contention cases, (defined in B.2.4 and B.2.5). It also leads to complexity of overlapped, concatenated and buffered interactions to optimize performance.

NOTE 2

The effects of the delay are factored out, not the delay itself.

NOTE 3

Synchronization communicated via a transit delay is necessarily only a pseudo-synchronization. The synchronization via the session-service creates an effect which is as if the actions in question occur at the same time.

1.5.2 Synchronization-Point Concept

Synchronization-points (defined in B.2.6) are the principal means of synchronization on a session-connection. They are constructed by use of certain services with service structure type 2 (defined in C.2.9), and non-destructive effects (defined in C.6).

The request event (defined in C.2.3, C.2.12) designates a point in the direction initiator to acceptor (defined in C.2.12 and C.2.13). The point is identified by a serial number (see 1.5.5). The response event (defined in C.2.5, C.2.12) which carries the same serial number signals the completion of synchronization.

Any semantics which ss users may give to their synchronization-points, or to the subunits of communication activity which they delimit (including check-pointing and commitment semantics) are transparent to the session service.

There are two main types of synchronization-point:

a) major-synchronization-point (defined in B.2.7),

b) minor-synchronization-point (defined in B.2.8).

1.5.3 Major-Synchronization-Point Concept

Figure 1.5/1 is a time-sequence diagram (notation defined in C.5), illustrating the main characteristics of a major-synchronization point.

The response/confirmation events (defined in C.2.5, C.2.6, C.2.12) define a time-slicing of activity on the session-connection, such that all communication before this division is isolated from all communication after it (there is one exception, see note 4).

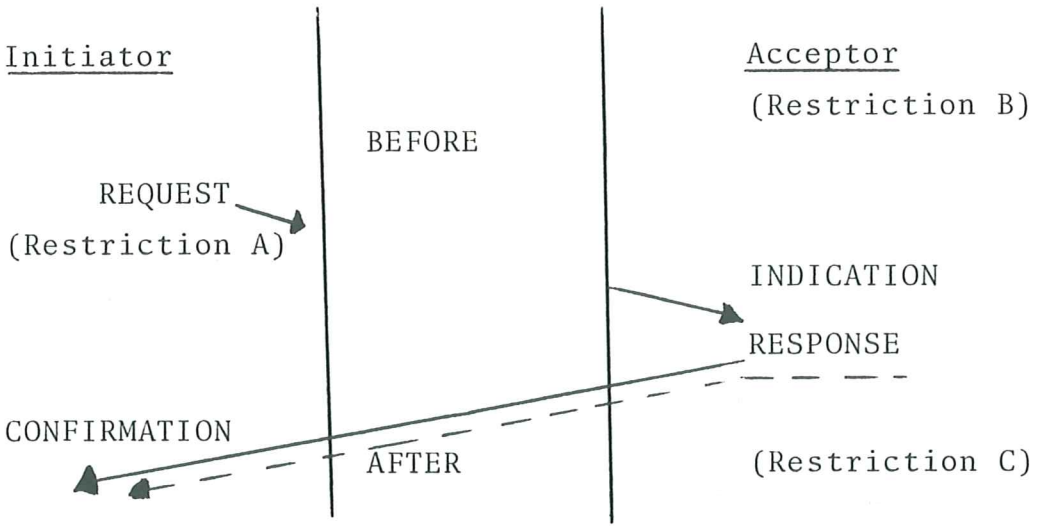


Fig. 1.5/1 - Major-Synchronization-Point

In order to ensure this absolute separation, the session service has the following characteristics:

- Restriction A: The initiator is not subsequently able to initiate any services until the response is received, except for requesting:

  . abnormal termination (see 1.5.9)
  . resynchronization (see 1.5.11).

- Restriction B: The acceptor is not able to initiate any service with type 2 service structure whose indication might arrive at the initiator of the major-synchronization-point request after the request event except for:

  . resynchronization.

- Restriction C: Expedited effects (see C.6) of services initiated by the acceptor after the response never arrive before the confirmation except for:

  . abnormal termination.

Restriction A ensures that the acceptor does not receive any old indication or confirmation events after the response event. Restriction B ensures that the initiator is not required to make response events in the interval between the request and confirmation events. The combination of the two restrictions prevents backward penetration by new events.

There is no restriction on other request and response events at the acceptor, or their corresponding indication and confirmation events at the initiator.

NOTE 4

The exception to these isolation provisions is that abnormal termination of the session connection may be initiated at any time.

1.5.4 Minor-Synchronization-Point Concepts

The role of a minor-synchronization-point is to define correlations between the flow of sequentially transmitted events in both directions. It does not define any correlation with expedited events.

Fig. 1.5/2 illustrates successive request/indication events defining boundaries between groups of events in the direction initiator to acceptor . Figure 1.5/3 illustrates the corresponding successive response/confirmation events defining boundaries between groups of other events in the direction acceptor to initiator. The combined correlation effect would be illustrated by superimposing figure 1.5/3 on figure 1.5/2, with figure 1.5/3 displaced along the time axis.

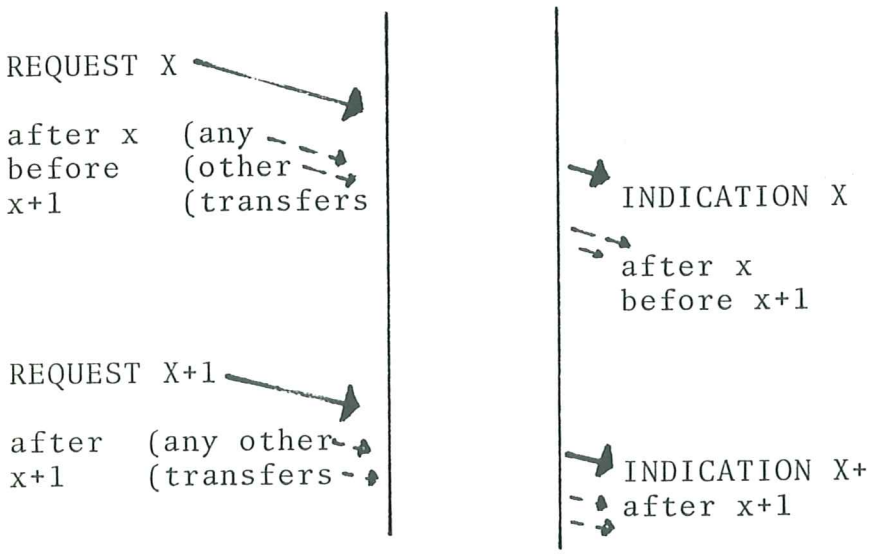


Fig. 1.5/2 Effects of Minor-Synchronization-Point Requests/ Indications.

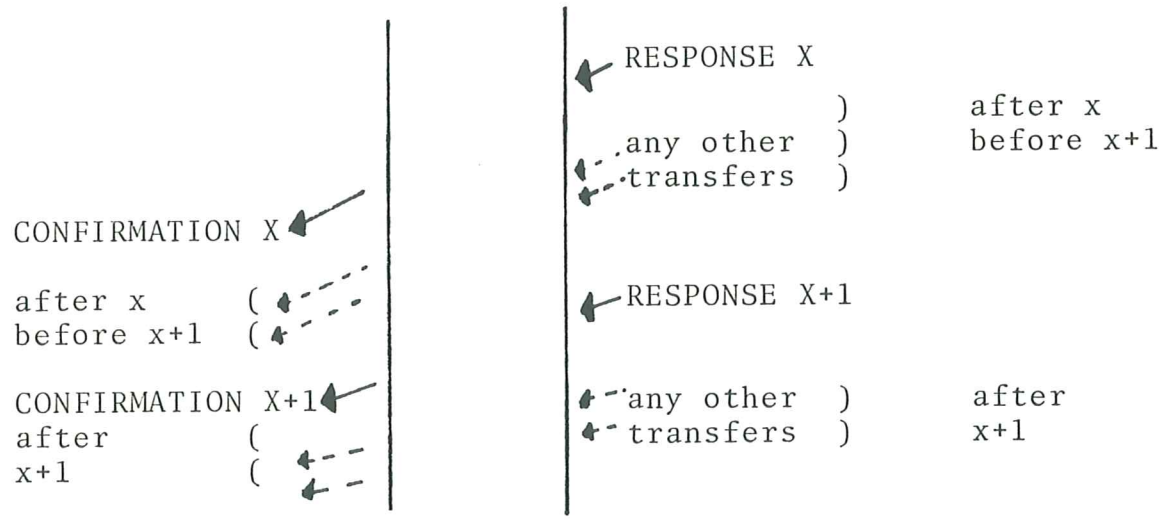


Fig. 1.5/3 Effects of Minor-Synchronization-Point Response/ Confirmations.

The initiator may make a simple synchronization structure by waiting for the confirmation event of each minor-synchronization-point before initiating further primitives. The synchronization structure then consists of a series of units of synchronization, one at a time.

The initiator may make more complex synchronization structures by issuing further primitives (e.g. for data transfers and synchronization points.) before receiving the confirmation event. The synchronization structure then consists of a series of concurrent units of synchronization. This is a performance optimization which might be characterized as "write ahead" or "pipelined operation".

There are two subtypes of minor-synchronization-point:

- normal,
- urgent.

The urgent subtype of minor-synchronization-point restricts the initiator not to initiate data transfers before the confirmation. It asks for the confirmation to be made "as soon as possible".

The normal subtype does not have this restriction. For example, it is possible to request a minor-synchronization-point while other minor-synchronization-points are unconfirmed. Confirmation of a subsequent synchronization-point implicitly confirms any previous unconfirmed synchronization points. This allows both synchronization structures described above.

The response/confirmation events of a minor-synchronization-point do not define a time-slicing of activity on the session connection. There is no assurance that all communication before is isolated from all communication afterwards. There are none of the controls itemized as in A, B and C in 1.5.3. This is illustrated in figure 1.5/4.

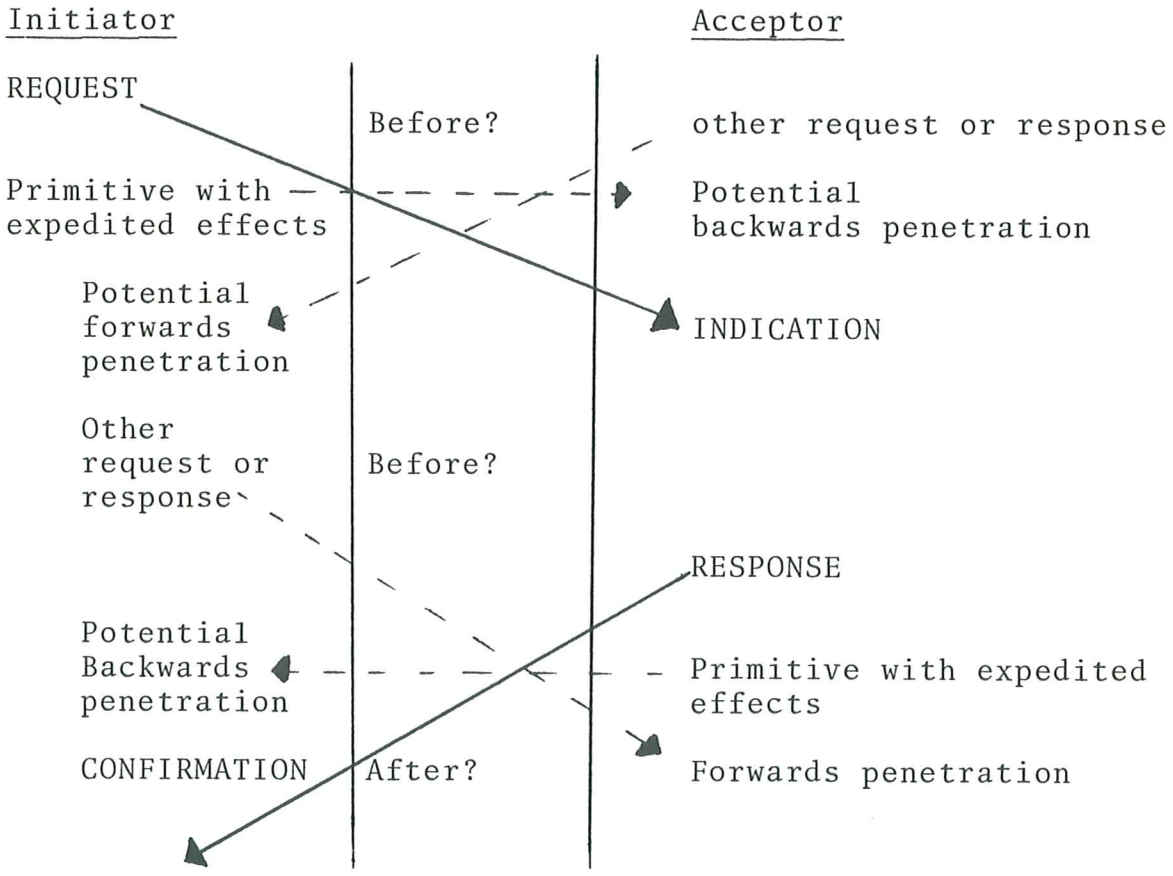


Fig. 1.5/4 - Minor-Synchronization-Point Penetration

1.5.5 Synchronization-Point-Serial-Number

The session-service provides an automatically incremented serial number to identify synchronization-points.

Certain services cause this value to be incremented. Typically the increment is by one (see note 5). Certain primitives may cause the current value to be changed to a new value possibly chosen by the ss-users. The range of this serial number is zero to 65535 (see note 6). No service increments beyond the maximum value (see note 7).

NOTE 5

In principle the increment may be variable and arbitrarily large, such that there are gaps in the serial numbers visible to ss-users. This characteristic ensures that application-entities do not depend on the assumption that the serial number (which is provided to them via the presentation-service) is incremented only as a result of their own actions. This gives freedom for the presentation-entities possibly to use the session-service to define additional synchronization-points for use internal to the presentation-layer.

NOTE 6

A larger range to accomodate the case when this maximum is not sufficient is for further study.

NOTE 7

The alternative of re-cycling through zero is an unnecessary complication if the range is large enough.

1.5.6 Dialogue-Unit Concepts

SS-users may structure the exchange of data between them into a series of dialogue-units (defined in B.2.10). The characteristic of a dialogue unit is that all communication within it is guaranteed to be isolated from all communication before and after it.

Figure 1.5/5 is a pictorial representation of this time slicing of a session-connection.

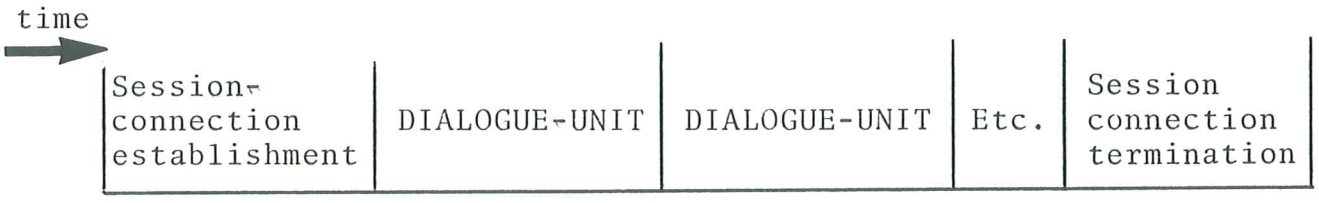


Fig. 1.5/5 - Dialogue-units

The beginning of the first dialogue-unit is implicit in the completion of session-connection establishment. Normal termination of a dialogue-unit is by means of a primitive which defines a major-synchronization-point. The

beginning of the next dialogue-unit is explicitly the termination of the current dialogue-unit (see note 8).

There is provision for destructive abnormal termination of a dialogue-unit by either ss-user. Connection termination also terminates the current dialogue-unit.

The session service does not constrain the semantics which ss-users may give to dialogue-unit synchronization structure. Any such semantics are transparent to the session-service.

NOTE 8

There is no recursivity of dialogue-units within dialogue-units. SS-users may, however, construct a recursive semantic structure delimited by means of the non-recursive synchronization structure.

1.5.7 Quarantining Concept

The quarantining concept is a means to optimize scheduling and buffering.

When quarantining is in use, an ss-user sending data delimits (explicitly or implicitly) a delivery-point (defined in B.2.16) when he requires the data to be delivered.

The delivery-point delimits a quarantine-unit (defined in ISO/DP 7498). None of the content of a quarantine-unit is delivered to the acceptor ss-user until the whole quarantine-unit has been transferred (see note 9).

The operation of quarantining is such that the acceptor ss-user has no information about delivery-points nor information that data has been discharged or grouped into quarantine-units.

NOTE 9

Use of this concept does not preclude implementations in which the content of a quarantine-unit is delivered to its user before it is complete, provided that the globally visible effects are as if none was delivered before completion.

1.5.8 Blocking

The blocking concept (defined in ISO/DP 7498) is included in the session service and protocol. The ss-user sending blocked data may use a local "push", provided at an SSAP in ways not defined by this Standard.

1.5.9 Session-Connection Termination Concepts

The session-service provides three types of session connection termination:

- abnormal termination (see note 10)
- non disruptive termination (see note 11)
- negotiated termination (see note 11).

Termination of the first type may cause loss of data. Termination of the second or third type causes no loss of data. The difference between the second and third type is that in the third type the acceptor may refuse the termination and continue the session-connection.

NOTE 10

This type relates to S-DISCONNECT and S-ABORT defined in 2.2.1.3 and 2.2.1.4.

NOTE 11

These types relate to S-RELEASE defined in 2.2.1.2.

1.5.10 Tokens Concept

Tokens (defined in B.2.11) provide a mechanism for ss-users to control the unambiguous dynamic and exclusive right to initiate certain functions.

Each token that is to exist during the lifetime of a session-connection is defined and initially assigned when the connection is established. It is assigned to one ss-user or the other. Thereafter, the assignment may be changed in various ways. Any token which is not defined during connection-establishment does not exist during the lifetime of the connection.

Four tokens are defined in this standard:

- data token (see B.2.12 and note 12),
- synchronize token (see B.2.13),
- end-DU token (see B.2.14),
- terminate token (see B.2.15).

Table 1.5/6 defines the effects that are dependent upon whether or not the above tokens are defined (i.e. used) for a session-connection.

Table 1.5/6 - Use of the Standard Tokens

Token	If defined	If not defined ....
Data token	TWA or one-way	TWS
Synchronize token	Minor-synchronization-points may be made	No minor-synchronization points
End-DU token	Major-synchronization-points may be made	No major-synchronization points
Terminate token	Termination may be negotiated	No negotiated termination

Table 1.5/7 defines the right conferred by assignment of the above tokens. It applies in the case where all the tokens are defined. If one or more tokens are not defined, the restriction defined in table 1.5/6 apply, and the entries for the not allowed actions and for the undefined token(s) are deleted from table 1.5/7, modifying the control effects.

Table 1.5/7 - Control Effects of Tokens

To initiate ...	It is necessary that these tokens be assigned (if defined) ...
Transfer of an SSDU	Data Token
Minor-synchronization-point	Data Token and Synchronize Token
Major-synchronization-point	Data Token and Synchronize Token and End-DU Token
Negotiated termination	Data Token and Terminate Token

For example, if the terminate token is defined and no others, communication is necessarily TWS only and there can be no synchronization-points (per table 1.5/6). Only the ss-user to whom the terminate token is currently assigned may initiate non-destructive negotiated termination, (per table 1.5/7) and either ss-user may send SSDUs (per table 1.5/6).

NOTE 12

The data token is equivalent to the "turn" concept which is a predecessor of the more general concept of tokens. The term "turn" is not used in this standard.

1.5.11 Resynchronization Concept

Resynchronization (defined in B.2.2) is a destructive change of synchronization, which may be initiated by either ss-user. It sets the session-connection to a defined state, and includes repositioning of tokens and setting the synchronization-point-serial-number to a new value. Resynchronization has a "purge" effect, and may cause loss of data.

Resynchronization abnormally terminates the current dialogue-unit. Two options are provided:

- abandon option,
- restart option.

These refer to the ss-user action relating to the work content of the current dialogue-unit and distinguish priority and different ways of setting the synchronization-point-serial-number.

1.5.12 Delimiter Hierarchy

There is a hierarchy of delimiter effects:

- at termination of the session-connection, the current dialogue-unit is terminated,
- termination of the current dialogue-unit implies a delivery-point in each direction, and implies confirmation of any outstanding minor-synchronization-points,
- any minor-synchronization-point implies a delivery-point,
- any token management action implies a delivery-point,
- a delivery-point implies end of SSDU.

2 Service

2.1 SERVICE OVERVIEW

The services defined by this standard are summarized in table 2.1/1. It lists the service primitives and groups them into five facilities.

The primitives are defined in 2.2.1. Their parameters are defined in 2.2.2.

Subsets of the primitives and parameters, which are oriented to particular uses of the session service, are defined in 2.3.

Table 2.1/1 - Structure of the Service Provided

Facility	Name of Primitive	
Session-connection establishment facility	S-CONNECT	request indication response confirmation
Session-connection termination facility	S-RELEASE	request indication response confirmation
	S-DISCONNECT	request indication
	S-ABORT	indication
Session-connection data transfer facility	S-DATA	request indication
	S-EXPEDITED	request indication
Session-connection quarantining facility	S-QUARANTINE-DELIVER	request
	S-QUARANTINE-CANCEL	request
Session-connection synchronization facility	S-SYNC	request indication response confirmation
	S-END-DU	request indication response confirmation
	S-RESYNC	request indication response confirmation
	S-TOKEN-GIVE	request indication
	S-PLEASE	request indication

2.2 SERVICE DESCRIPTION

The service is described by using the service description notation and terminology defined in Appendix C which should be read at this point.

2.2.1 Primitives

NOTE 13

Use of certain primitives defined in this subclause is precluded in particular subsets (see 2.3).

2.2.1.1 S-CONNECT primitives

The S-CONNECT primitives are the means for two ss-users to establish a session-connection between them.

The effects are sequentially transmitted and non-disruptive. The service structure is type 2. Parameters are defined in 2.2.2.1. The right to be initiator is not subject to token assignments.

The outcome is either of:

- the session-connection is successfully established and ready for use,
- the session-connection is rejected and its effects are discarded by the session service.

2.2.1.2 S-RELEASE primitives

The S-RELEASE primitives are the means for negotiated termination (defined in 1.5.9) of a session-connection (see note 14).

The effects are sequentially transmitted and non-disruptive. The service structure is type 2. Parameters are defined in 2.2.2.2. The right to be initiator may be subject to the current token assignments, as defined in table 1.5/6 and 1.5/7.

The outcome is either of:

- the acceptor ss-user agrees and the session-connection is terminated,
- the acceptor ss-user does not agree, and the session-connection is not terminated.

NOTE 14

Termination is not negotiated if the terminate token is not defined (see table 1.5/6), in which case the termination is of the basic non-destructive type (defined in 1.5.9).

2.2.1.3 S-DISCONNECT primitives

The S-DISCONNECT primitives are the means for abnormal termination (defined in 1.5.9) of a session-connection, by either ss-user acting unilaterally.

The effects are disruptive. The service structure is type 1. Parameters are defined in 2.2.2.3. The right to be initiator is not subject to token assignments.

2.2.1.4 S-ABORT primitive

The S-ABORT primitive is the means for abnormal termination (defined in 1.5.9) of a session-connection by the session service.

The effects are disruptive. The service structure is type 3. Parameters are defined in 2.2.2.4. This primitive is not affected by token assignments.

2.2.1.5 S-DATA primitives

The S-DATA primitives are the means for an ss-user to transfer an SSDU to the other ss-user.

The effects are sequentially transmitted and non-disruptive. The service structure is type 1. Parameters are defined in 2.2.2.5. The right to be initiator may be subject to the current token assignments, as defined in tables 1.5/6 and 1.5/7.

2.2.1.6 S-EXPEDITED primitives

The S-EXPEDITED primitives are the means for an ss-user to transfer an expedited SSDU of limited size to the other ss-user.

The effects are expedited and non-disruptive. The service structure is type 1. Parameters are defined in 2.2.2.6. The right to be initiator is not subject to token assignments.

2.2.1.7 S-QUARANTINE-DELIVER primitive

The S-QUARANTINE-DELIVER primitive is the means for an ss-user to cause release of previous indication events which were originated by the initiator ss-user and may be subject to quarantining.

The effects are sequentially transmitted and non-disruptive. The service structure is type 4. Parameters are defined in 2.2.2.7. The right to be initiator may be subject to the current token assignments, as defined in tables 1.5/6 and 1.5/7.

2.2.1.8 S-QUARANTINE-CANCEL primitive

The S-QUARANTINE-CANCEL primitive is the means for an ss-user to cause discard of previous indication events which were originated by the initiator ss-user and are subject to quarantining.

The effects are sequentially transmitted and disruptive. The service structure is type 4. Parameters are defined in 2.2.2.8. The right to be initiator may be

subject to the current token assignments, as defined in tables 1.5/6 and 1.5/7.

The outcome is that the indication events concerned do not occur and all their effects are discarded by the session service.

2.2.1.9 S-SYNC primitives

The S-SYNC primitives are the means for an ss-user to define a minor-synchronization-point, and for the acceptor ss-user to confirm this synchronization.

The effects are sequentially transmitted and non-disruptive. The service structure is type 2. Parameters are defined in 2.2.2.9. The right to be initiator is subject to the current token assignments, as defined in tables 1.5/6 and 1.5/7.

NOTE 15

The acceptor ss-user may reject the proposed synchronization by initiating an S-RESYNC primitive.

2.2.1.10 S-END-DU primitives

The S-END-DU primitives are the means for an ss-user to define a major-synchronization-point which terminates the current dialogue-unit normally (see note 16) and for the acceptor ss-user to confirm this synchronization (see note 17).

The effects are sequentially transmitted and non-disruptive. The service structure is type 2. Parameters are defined in 2.2.2.10. The right to be initiator is subject to the current token assignments, as defined in tables 1.5/6 and 1.5/7.

NOTE 16

The dialogue-unit is not actually terminated until the response/confirmation events.

NOTE 17

The acceptor ss-user has no means to reject termination of the dialogue unit, but may force a different termination by initiating an S-RESYNC.

2.2.1.11 S-RESYNC primitives

The S-RESYNC primitives are the means for an ss-user to reset the session-connection, and to agree associated parameters with the acceptor ss-user. If the dialogue-unit concept is in use, the current dialogue unit is terminated. This is referred to as abnormal termination of the dialogue unit.

The effects are disruptive. The service structure is type 2. Parameters are defined in 2.2.2.11. The right to be initiator is not subject to token assignments.

2.2.1.12 S-TOKENS-GIVE primitives

The S-TOKENS-GIVE primitives are the means for an ss-user to assign to the acceptor ss-user one or more tokens currently assigned to the initiator.

The effects are sequentially transmitted and non-disruptive. The service structure is type 1. Parameters are defined in 2.2.2.12.

2.2.1.13 S-PLEASE primitives

The S-PLEASE primitives are the means for an ss-user to ask the other ss-user to give to him (the initiator) one or more tokens which are not currently assigned to him.

The effects are sequentially transmitted and non-disruptive. The service structure is type 1. Parameters are defined in 2.2.2.13.

The acceptor ss-user has the choice to ignore the advice or act upon it.

2.2.2 Parameters

The notation used in this subsection is defined in C.7 which should be read at this point.

NOTE 18

Use of certain parameters and parameter values defined in this subclause is precluded in particular subsets (see 2.3).

2.2.2.1 Parameters of S-CONNECT

The parameters of the S-CONNECT primitives and their values are defined in table 2.2/1 and the following rules which reference it.

Table 2.2/1 - Parameters of S-CONNECT

PARAMETER NAMES	Request	Indication	Response	Confirmation
Result	X	X	B1	U
Connection-identifier	B2	U	B3	U
Initiator address	D4	U5	X	X
Acceptor address	D4	U5	B6	U5
Subset choice	D7	U	X8	X
Subset parameters	-9	-9	-9	-9
Max. SSDU size I to A	D10	U	D10	U
Max. SSDU size A to I	D11	U	D11	U
SS-user data	D12	U	D12	U
Reason code	X	X	B13	U

Rule 1 If the connection is accepted by the acceptor ss-user, the value is "accept". Otherwise, the value is "reject". In this case, all the other parameters of the S-CONNECT response and confirmation have the "null" values, except possibly the ss-user data and the reason code.

Rule 2 This value is the initiator's proposal for the session-connection-identifier. Its structure and format and origin are not defined in this standard. The source code B allows the value to be originated by the ss-user or by the session service or by the two acting together. The maximum size is 16 octets.

Rule 3 This value is the agreed session-connection-identifier. It may be the same value as the initiator's proposal, or different. It may be a concatenation of an acceptor's component with the initiator's proposal. Otherwise as Rule 2.

Rule 4 Session-service-address, identifying a session-service-access-point (SSAP) in ways not defined in this standard. But the maximum size is 16 octets.

Rule 5 Where there is address translation within the session service, this value may be different from that in the preceding event. Otherwise as Rule 4.

Rule 6 Session-service-address, identifying a session-service-access-point (SSAP) in ways not defined in this standard. But the maximum size is 16 octets. Where there is generic addressing or re-direction within the session service, this value may be different from that in the indication event.

Rule 7 This value identifies any one of the subsets defined in 2.3.

Rule 8 The value in the request/indication is not negotiable.

Rule 9 The rules for these subset parameters depend on which subset is chosen. See 2.3.

Rule 10 Value in the request/indication primitives is the initiator requirement about the maximum size of SSDU in the direction initiator to acceptor. Value in the response/confirmation primitives is the acceptor capability about that same size. Both values may be "null". The session service imposes no relationship between the values in the request/indication primitives and the value in the response/confirmation primitives.

Rule 11 Same as Rule 10, but invert the terms "requirement" and "capability" and the direction is now acceptor to initiator.

Rule 12 SS-user data content is transparent to the session service. Size is an integral number of octets. Minimum size is zero octet. Maximum size is 64 octets.

Rule 13 Present only if result is "reject". Gives session-service reason for not accepting the session connection. See 2.2.3.

2.2.2.2 Parameters of S-RELEASE

The parameters of the S-RELEASE primitives and their values are defined in table 2.2/2 and the following rules which reference it.

Table 2.2/2 - Parameter of S-RELEASE

PARAMETER NAMES	Request	indication	response	confirmation
Result	X	X	D1	U
SS-user data	D2	U	D2	U

Rule 1. Value is "affirmative" if the acceptor ss-user agrees to the connection being terminated. Otherwise it is "negative" (see note 19).

Rule 2. SS-user data is transparent to the session service. The size is an integral number of octets. The minimum size is zero octet. The maximum size is 32 octets.

NOTE 19

The choice to give the response "negative" is not available if the termination-token is not defined (i.e. the termination is not negotiated). See 1.5.10.

2.2.2.3 Parameters of S-DISCONNECT

The parameters of the S-DISCONNECT primitives and their values are defined in table 2.2/3 and the following rule which references it.

Table 2.2/3 - Parameters of S-DISCONNECT

PARAMETER NAMES	request	indication
SS-user data	D1	U

Rule 1. The ss-user data is transparent to the session service. The maximum size is three octets.

2.2.2.4 Parameters of S-ABORT

The parameters of the S-ABORT primitive and their values are defined in table 2.2/4 and the following rule which references it.

Table 2.2/4 - Parameters of S-ABORT

PARAMETER NAMES	Indication	Indication
Reason code	U1	U1

Rule 1. See 2.2.3. This parameter may have different values at the two SSAPs in circumstances where the reason is perceived to be different at the two ends of the session-connection.

2.2.2.5 Parameters of S-DATA

The parameters of the S-DATA primitives and their values are defined in table 2.2/5 and the following rule which references it.

Table 2.2/5 - Parameters of S-DATA

PARAMETER NAMES	Request	Indication
SSDU	D1	U

Rule 1. The SSDU is transparent to the session service. The size is an integral number of octets. The minimum size is zero octet. There is no maximum size restriction, except that derived cumulatively from the corresponding maximum quarantine size parameters of the S-CONNECT primitives.

2.2.2.6 Parameters of S-EXPEDITED

The parameters of the S-EXPEDITED primitives and their values are defined in table 2.2/6 and the following rule which references it.

Table 2.2/6 - Parameters of S-EXPEDITED

PARAMETER NAMES	Request	Indication
XSSDU	D1	U

Rule 1. The XSSDU is transparent to the session service. Size is an integral number of octets between zero and six.

2.2.2.7 Parameters of S-QUARANTINE-DELIVER

None.

2.2.2.8 Parameters of S-QUARANTINE-CANCEL

None.

2.2.2.9 Parameters of S-SYNC

The parameters of the S-SYNC primitives and their values are defined in table 2.2/7 and the following rules which reference it.

Table 2.2/7 - Parameters of S-SYNC

PARAMETER NAMES	Request	Indication	Response	Confirmation
Type	D1	U	X	X
Serial Number	U2	U	D3	U
SS-user-data	D4	U	D4	U

Rule 1. Type is "normal" or "urgent" (see 1.5.4).

Rule 2. Synchronization-point-serial-number, incremented as defined in 1.5.5.

Rule 3. This value identifies a previously indicated but not yet confirmed minor-synchronization-point, which is now confirmed (see 1.5.4).

Rule 4. The ss-user-data is transparent to the session service. The maximum size is 6 octets.

2.2.2.10 Parameters of S-END-DU

The parameters of the S-END-DU primitives and their values are defined in table 2.2/8 and the following rules which reference it.

Table 2.2/8 - Parameters of S-END-DU

PARAMETER NAMES	Request	Indication	Response	Confirmation
Serial Number	U1	U	X2	X
ss-user-data	D3	U	D3	U

Rule 1. Synchronization-point-serial-number, incremented as defined in 1.5.5.

Rule 2. The response/confirmation events necessarily relate to the same serial number as the request/indication events.

Rule 3. The ss-user-data is transparent to the session service. The maximum size is 6 octets.

2.2.2.11 Parameters of S-RESYNC

The parameters of the S-RESYNC primitives and their values are defined in table 2.2/9 and the following rules which reference it.

Table 2.2/9 - Parameters of S-RESYNC

PARAMETER NAMES	Request	Indication	Response	Confirmation
Resync type	D1	U	X	X
Serial Number	X2, D3	X2, U3	U2, X3	U2, X3
ss-user-data	D4	U	D4	U
Data token	D5	U	D6	U
Sync. token	D5	U	D6	U
End DU token	D5	U	D6	U
Termin. token	D5	U	D6	U

Rule 1. This value defines the ss-user action relating to the work content of the dialogue unit which is terminated. The value is either "abandon" or "restart". If there is contention between S-RESYNC requests "abandon" has priority (see note 20).

Rule 2. If the resync type in the request is "abandon", this rule applies. The value is provided by the session-service (see note 21) incremented as defined in 1.5.5.

Rule 3. If the resync type in the request is "restart" this rule applies. The value is provided by the initiator and is not negotiable (see note 20). It is not less than the value at the start of the current dialogue-unit.

Rule 4. The ss-user-data is transparent to the session service. The maximum size is 6 octets.

Rule 5. If the token is not defined for this connection the value is "null", otherwise it is "initiator" or "acceptor" or "acceptor chooses".

Rule 6. If the value in the request/indication is "acceptor chooses" this value is either "initiator" or "acceptor". Otherwise it is the same.

NOTE 20

Negotiation by means of multiple use of the S-RESYNC primitives and detailed contention resolution are defined in B.2.5.2.

NOTE 21

The value is only known at the time of the response/confirmation events because it depends on the outcome of protocol exchanges inside the session layer.

2.2.2.12 Parameters of S-TOKENS-GIVE

The parameters of the S-TOKENS-GIVE primitive and their values are defined in table 2.2/10 and the following rule which reference it.

Table 2.2/10 - Parameters of S-TOKENS-GIVE

PARAMETER NAMES	Request	Indication
Token given	D1	U

Rule 1. The value is "data token" or "synchronize token", or "end-DU-token", or "terminate token" or any combination of these.

2.2.2.13 Parameters of S-PLEASE

The parameters of the S-PLEASE primitives and their values are defined in table 2.2/11 and the following rule which references it.

Table 2.2/11 - Parameters of S-PLEASE

PARAMETER NAMES	Request	Indication
Token asked for	D1	U
ss-user data	D2	U

Rule 1. Value is "data token" or "synchronize token", or "end-DU-token", or "terminate token", or any combination of these.

Rule 2. The ss-user data is transparent to the session service. Maximum size is 1 octet.

2.2.3 Reason Codes

Reason codes indicate session service generated information that qualifies the inability of the session service to perform the service concerned. The values are defined in 3.4.

SS-user-data semantic and encoding are transparent to the session service. The session service may limit the size of ss-user-data. The actual limitations are indicated for each primitive (see 2.2.2).

2.3 SERVICE SUBSETS

2.3.1 General

Subsets of the session service are defined to achieve simplification and variety control. Each subset is a

complete set of services for establishing, using and terminating a session-connection.

Four subsets are defined. Each is identified by a letter code and a name:

Subset A - basic subset
Subset B - basic interactive subset
Subset C - basic synchronized subset
Subset D - basic TWA subset

2.3.2 Subset A: Basic Subset

2.3.2.1 Purpose

Subset A provides to ss-users only the means of data transfer. Any structuring of the interaction is wholly an ss-user responsibility.

2.3.2.2 Content

The service-primitives included in this subset are:

- S-CONNECT primitives,
- S-RELEASE primitives,
- S-DISCONNECT primitives,
- S-ABORT primitive,
- S-DATA primitives,
- S-EXPEDITED primitives.

The selection of all these primitives and their associated services is implicit in the choice of this subset (i.e. value "A" in the subset choice parameter of S-CONNECT, see 2.2.2.1). There is no parameter in S-CONNECT which is specific to this subset.

No tokens are defined. By application of the rules in table 1.5/6 interaction type is TWS and negotiated termination is not available in this subset. The termination via S-RELEASE is only non-destructive and the S-RELEASE result code parameter value is always "affirmative".

2.3.3 Subset B: Basic Interactive Subset

2.3.3.1 Purpose

Subset B provides to ss-users an organized dialogue structure with selection of the interaction type (i.e. TWS/TWA).

2.3.3.2 Content

The service primitives included in this subset are:

- S-CONNECT primitives,
- S-RELEASE primitives,
- S-DISCONNECT primitives,
- S-ABORT primitive,
- S-DATA primitives,
- S-EXPEDITED primitives,

- S-QUARANTINE-DELIVER primitive,
- S-QUARANTINE-CANCEL primitive,
- S-TOKENS-GIVE primitives,
- S-PLEASE primitives.

The selection of all these primitives and their associated services is implicit in the choice of this subset (i.e. value "B" in the subset choice parameter of S-CONNECT, see 2.2.2.1).

Interaction type is selected at connection establishment (i.e. the data-token is or is not defined and assigned). Negotiated termination is supported (i.e. the terminate-token is always defined and assigned). No other tokens are defined.

The additional parameters in S-CONNECT which are specific to this subset and their values are defined in table 2.3/1 and the following rules which reference it.

Table 2.3/1 - S-CONNECT, parameters specific to Subset B

PARAMETER NAMES	Request	Indication	Response	Confirmation
Existence of data token	D1	U	D2	U
Data token	D3	U	D4	U
Terminate token	D5	U	D6	U
Quarantining max size I to A	D7	X	X	U7
Quarantining max size A to I	X	U8	D8	X

Rule 1. Value is "defined" or "not defined".

Rule 2. Value is same as in request/indication

Rule 3. Value is "initiator", or "acceptor", or "acceptor chooses". This parameter is only present if the value of the existence of Data token parameter is "defined".

Rule 4. If the value in the request/indication is "acceptor chooses", this value is either "acceptor" or "initiator". Otherwise the assignment in the request/indication is not negotiable, and this value is the same. This parameter is only present if the value of the existence of Data token parameter is "defined".

Rule 5. Value is "initiator" or "acceptor" or "acceptor chooses".

Rule 6. If the value in the request/indication is "acceptor chooses", this value is either "acceptor" or "initiator". Otherwise the assignment in the request/indication is not negotiable, and this value is the same.

Rule 7. The value in the request primitive indicates the initiator requirement for the maximum size of a quarantine unit in the direction initiator to acceptor. Value "null" means no quarantining requirement. The value in the confirmation primitive indicates the service capabilities. The value in the confirmation shall be greater than or equal to the value in the request. This parameter is not visible to the acceptor.

Rule 8. The value in the indication primitive indicates the service capabilities for the maximum size of a quarantine unit in the direction acceptor to initiator. Value "null" means no quarantining capability. The value in the response primitive indicates the acceptor requirement. The value in the response shall be lower or equal than the value in the request. This parameter is not visible to the initiator.

2.3.4 Subset C: Basic Synchronized Subset

2.3.4.1 Purpose

Subset C provides to ss-users a synchronized and organized dialogue structure with selection of the interaction type (i.e. TWS/TWA).

2.3.4.2 Content

The service primitives included in this subset are:

- S-CONNECT primitives,
- S-RELEASE primitives,
- S-DISCONNECT primitives,
- S-ABORT primitive,
- S-DATA primitives,
- S-SYNC primitives,
- S-END-DU primitives,
- S-RESYNC primitives,
- S-TOKENS-GIVE primitives,
- S-PLEASE primitives.

The selection of all these primitives and their associated services is implicit in the choice of this subset (i.e. value "C" in the subset choice parameter of S-CONNECT, see 2.2.2.1).

Interaction type is selected at connection establishment (i.e. the data-token is or is not defined and assigned). Major and minor synchronization-points are supported (i.e. the synchronize-token and end-DU-token are always defined and assigned). Negotiated termination is suported (i.e. the terminate-token is always defined and assigned).

The additional parameters in S-CONNECT which are specific to this subset and their values are defined in table 2.3/2 and the following rules which reference it.

Table 2.3/2 - S-CONNECT, Parameters Specific to Subset C

PARAMETER NAMES	Request	Indication	Response	Confirmation
Existence or data token	D1	U	D2	U
Data token	D3	U	D4	U
Synchronize token	D5	U	D6	U
End DU token	D5	U	D6	U
Terminate token	D5	U	D6	U
Initial serial number	D7	U	D8	U
Blocking I to A	D9	X	X	X
Blocking A to I	X	X	D10	X

Rule 1. Value is "defined" or "not defined".

Rule 2. Value is same as in the request/indication.

Rule 3. Value is "initiator" or "acceptor" or "acceptor chooses". This parameter is only present if the value of the existence of Data token parameter is "defined".

Rule 4. If the value in the Request/indication is "acceptor chooses" this value is either "initiator" or "acceptor". Otherwise the assignment in the request/indication is not negotiable and this value is the same. This parameter is only present if the value of the existence of Data token parameter is "defined".

Rule 5. Value is "initiator" or "acceptor" or "acceptor chooses".

Rule 6. If the value in the request/indication is "acceptor chooses" this value is either "initiator" or "acceptor". Ohterwise the assignment in the request/indication is not negotiable and this value is the same.

Rule 7. This is any value in the range defined in 1.5.5.

Rule 8. This value takes precedence if it is different from that in the request/indication. It is any value in the range defined in 1.5.5.

Rule 9. Requests blocking optimization of SSDU transfers in the direction initiator to acceptor. The value is "yes" or "no". This information is used inside the session service and is not visible to the other ss-user.

Rule 10. As rule 9, except direction is acceptor to initiator.

2.3.5 Subset D: Basic TWA Subset

2.3.5.1 Purpose

Subset D provides to ss-users a simplified TWA interaction structure.

2.3.5.2 Content

The service primitives included in this subset are:

- S-CONNECT primitives,
- S-RELEASE primitives,
- S-DISCONNECT primitives,
- S-ABORT primitives,
- S-DATA primitives,
- S-TOKENS-GIVE primitives,
- S-PLEASE primitives.

The selection of all these primitives and their associated services is implicit in the choice of this subset (i.e. value "D" in the subset choice parameter of S-CONNECT see 2.2.2.1). Interaction type is TWA (i.e. the Data token is always defined and assigned). Not other tokens are defined. By application of the rules in table 1.5/6, interaction type is TWA and negotiated termination is not available in this subset. The termination via S-RELEASE is only non-destructive and the S-RELEASE result code parameter value is always "affirmative".

The additional parameters in S-CONNECT which are specific to this subset and their values are defined in table 2.3/3 and the following rules which reference it.

Table 2.3/3 - S-CONNECT, parameters specific to subset D

PARAMETER NAMES	Request	Indication	Response	Confirmation
Data token	D1	U	D2	U

Rule 1. Value is "initiator" or "acceptor" or "acceptor chooses".

Rule 2. If the value in the request/indication is "acceptor chooses", this value is either "acceptor" or "initiator". Otherwise the assignment in the request/indication is not negotiable, and this value is the same.

3 Protocol

3.1 PROTOCOL OVERVIEW

3.1.1 Model of the Layer

The specification in this standard is constructed using a particular conceptual model of the session layer. This subclause describes that model.

The model distinguishes two different aspects of session layer protocol:

- The rules by which session protocol data units (SPDUs) are generated, exchanged and accepted or rejected: this is called the session layer SPDU protocol (see clause 3.2).
- The rules by which the transport service is used and manipulated: this is called the session layer transport service mapping (TSM) protocol (see 3.3).

The two protocols above are intimately related to each other, and inherently have a common locus in a protocol machine at each session entity. This is called the session protocol machine (SPM).

Figure 3.1/1 summarizes this model of the layer.

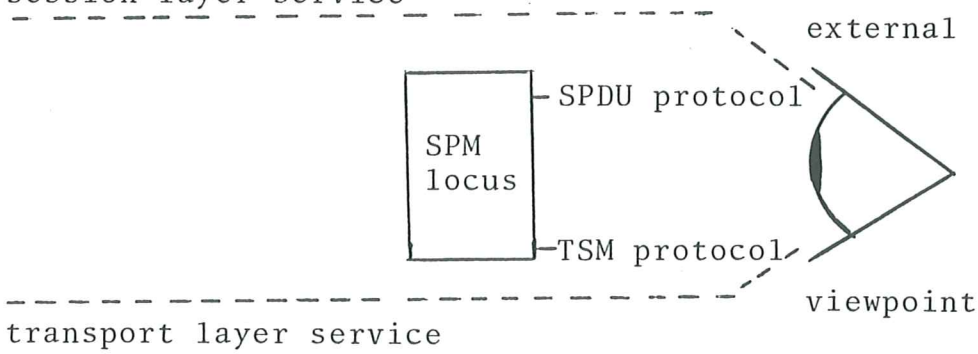


Fig.3.1/1 The criterion of external visibility applied to the Model of the Layer

The specification in this standard is concerned with the externally visible characteristics of the session layer, i.e. the SPM behaviour and its protocols. Internal functions such as address translation, title to address mapping, buffering, queuing and the detailed manifestation of the abstract layer service events defined in the layer service standards are outside its scope (see 1.2).

Another important characteristic of this model is that the detailed division of functionality between the SPDU protocol and the TSM protocol is arranged in such a way that the specification of the SPDU protocol is invariant to transport related variables. Such variability is localized in the specification of the TSM protocol.

NOTE 22

The conceptual partitioning of functions which is described in this model is for purpose of modular description and specification. No equivalent partitioning is required in implementations.

3.1.2 Specification of the session protocol

The protocol is described by means of events, states and transitions. Subsection 3.2 provides a narrative description. Appendix D provides the formal description.

3.2 SPDU PROTOCOL

3.2.1 General

This is a narrative description of the protocol. The protocol is described message by message. For each message type, there is a brief description under the headings Function, Content, Sending and Receiving, and sometimes Additional Information. The description references the message contents defined in 3.4 and the states and events defined in Appendix D. It describes the more usual valid sequences of messages. It does not describe the unusual interactions.

For sending, the validity of the associated SPM Service request and response events depends upon the current SPM state, and other conditions which are described in each subclause and in Appendix D.

For receiving, the valid conditions are more complex because they are the product of the valid sending conditions and the current state of the receiving SPM etc., plus the variable effects of the transport service delay characteristics. Therefore, only brief details are given in this part of the description.

The message types are described in four groups:

- Connection protocol
- Disconnection protocol
- Data Transfer protocol
- Synchronization protocol.

3.2.2 Connection Protocol

Session connection protocol is used:

- to establish a session connection between two users of the Session Layer,
- to negotiate session connection parameters,
- to transfer user data.

Session connection establishment protocol uses the CONNECT (CN), REFUSE (RF) and ACCEPT (AC) messages.

3.2.2.1 The CONNECT message

Function

The CONNECT (CN) message is used to request the establishment of a session connection and proposes parameter values.

Content

The message contains the description of the global capabilities of the sender:

- Protocol Identification parameters. These identify the Session Layer protocol as being the ECMA Session Protocol version 1. They nominate also the subset being used.
- Dialogue Negotiation parameters:
  - mode of dialogue (use of tokens)
  - initial tokens setting (initiator, acceptor or acceptor chooses)
  - maximum quarantine unit size requirement initiator to acceptor
  - maximum quarantine unit size capability acceptor to initiator
  - SSDU maximum size requirement initiator to acceptor
  - SSDU maximum size capability acceptor to initiator
  - SPDU maximum size requirement initiator to acceptor after connection establishment (minimum value is 36 octets)
  - SPDU maximum size capability acceptor to initiator after connection establishment (minimum value is 36 octets)
  - serial number to be used for the first mark.
  - initiator blocking requirement (value "yes" or "no")
  - initiator deblocking capability (value "yes" or "no").
- Addressing parameters (carried transparently, may be omitted):
  - the session layer address of the initiator ss-user
  - the session layer address of the acceptor ss-user
  - the connection-identifier.
- ss-user data (limited to 64 octets).

Sending

A valid "SPM service ESTABLISH request" event (EVE1), results in a CONNECT (CN) message. Functions within the layer but external to the SPM determine the transport address of the session entity supporting the destination user, so that a transport connection is provided to the appropriate destination. The CONNECT message is sent on the transport normal flow. The SPM goes to the state "waiting for ACCEPT message" (STA 2A).

Receiving

A valid "incoming CONNECT message" event (EVE 11) normally results in an "SPM service establish indication" event (EVE 101). The session service user is selected according to the destination address parameter of the CONNECT message. The standard subset selection and dialogue negotiation parameters are matched against the values that the session entity can accept.
In particular, it is checked whether:

- the proposed SPDU size is acceptable,
- the maximum quarantine unit size requirements in the direction intiator to acceptor is less or equal to the maximum quarantining capability of the receiving SPM,
- the blocking requirement matches with the deblocking capability of the receiving SPM.

The user data is passed to the session service user. The SPM goes to the state "waiting for ESTABLISH response" (STA 8).

Additional Information

The sender of this message is defined to be the "contention-winner" for the whole duration of the session connection.

3.2.2.2 The ACCEPT message

Function

The ACCEPT message (AC) is used to confirm successful establishment of a session connection, in response to a CONNECT message.

Content

The structure of this message is the same as the CONNECT message (see 3.2.2.1, Content). The dialogue negotiation parameters values follow the rules defined below.

- mode of dialogue (values same as in CONNECT),
- token setting (values same as in CONNECT except when it was "acceptor chooses" where value shall be either "initiator" or "acceptor",
- maximum quarantine unit size requirement acceptor to initiator (value less than or equal to initiator capability),
- maximum quarantine unit size capability initiator to acceptor (value greater than or equal to the initiator requirement),
- SSDU maximum size requirement acceptor to initiator (value unrestricted),

- SSDU maximum size capability initiator to acceptor (value unrestricted),
- SPDU maximum size requirement acceptor to initiator after connection establishment (value less than or equal to the initiator capability, but greater than 36 octets),
- SPDU maximum size capability initiator to acceptor after connection establishment (value greater than or equal to the initiator requirement),
- serial number to be used for the first mark (no relationship to the same parameter in the CONNECT),
- acceptor blocking requirement (value "yes" only if initiator deblocking capability had value "yes", otherwise value "no"),
- acceptor deblocking capability (value "yes" if initiator blocking requirement had value "yes" otherwise value "yes" or "no").

Sending

A valid "SPM service ESTABLISH response" event (EVE 25), results in an ACCEPT message. The message is sent on the transport normal flow. The SPM goes to the state "DATA TOKEN my side (STA 7)" or "DATA TOKEN not my side (STA 13)" depending upon the initial chosen position of the data token. The SPM sets its internal variable V(M), which contains the serial number to be used for the next type B or C or D mark, (see 3.2.4.1) to the value given by the ss-user (parameter EVE 25).

Receiving

A valid "incoming ACCEPT message" event (EVE 12) normally results in an "SPM service ESTABLISH confirmation" event (EVE 112). The user data is passed to the session service user. The SPM goes to the state "DATA TOKEN my side (STA 7)" or "DATA TOKEN not my side (STA 13)" depending on the chosen initial position of the data token. The SPM sets its internal variable V(M), which contains the serial number to be used for the next type B or C or D mark, to the value contained in the ACCEPT message.

Additional Information

The values of the maximum SPDU size and maximum quarantine unit size after sending and/or receiving the ACCEPT message are as follows:

- in the direction initiator to acceptor: equal to the requirement of the initiator,
- in the direction acceptor to initiator: equal to the requirement of the acceptor.

In a similar way, the blocking function is "on" according to the requirement of the initiator and the accept-

or. The serial number value to be used for the first mark is the value contained in the ACCEPT message. After the successful establishment of the session connection, the first DIALOG-UNIT is implicitly entered.

3.2.2.3 The REFUSE message

Function

The REFUSE message (RF) is used to reject the establishment of a session connection.

Content

The message contains:

- protocol identification,
- a reason code,
- transparent ss-user data (limited to 64 octets)
- a code specifying transport disconnection.

Sending

A valid "SPM service REJECT request" event (EVE 10A) results in a REFUSE message. The message is sent on the transport normal flow. Status information relating to the connection is cleared (i.e. the connection ceases to exist). The SPM goes to the state "unconnected" (STA 1).

Receiving

A valid "incoming REFUSE message" event (EVE 12A) normally results in an "SPM service REJECT indication" event (EVE 110A). The receiving session entity does not respond. Status information relating to the connection is cleared (i.e. the connection ceases to exist). The SPM goes to the state "unconnected" (STA 1).

3.2.3 Disconnection Protocol

Session disconnection protocol is used to terminate a session connection. It uses the FINISH (FN), DISCONNECT (DN), NOT FINISHED (NF) and ABORT (AB) messages.

3.2.3.1 The FINISH message

The FINISH message (FN) is used to initiate orderly termination of a session connection. The FINISH message seeks a DISCONNECT message as a response to complete the termination of the connection.

Content

The message may contain transparent ss-user data (up to 32 octets).

Sending

The right to issue a FINISH message is regulated by the TERMINATE TOKEN.
A valid "SPM service RELEASE request" event (EVE 3) results in a FINISH message. The message is sent on the transport normal flow.
The SPM goes to the state "waiting for DISCONNECT message" (STA 3), in which the ss-user shall not send on the expedited flow or on the normal flow, but is able to receive data on both flows.

Receiving

A valid "incoming FINISH message" event (EVE 13) normally results in an "SPM service RELEASE indication" event (EVE 103). The user data is passed to the session service user. The SPM goes to the state "waiting for RELEASE response" event (STA 9).

Additional information

The expected outcome at the receiving end is an "SPM service RELEASE affirmative response" event (EVE 26A). This results in a DISCONNECT message being sent.

Alternatively, the receiving session service user may wish to continue data transfer. The outcome is then a "SPM service RELEASE negative response" (EVE 26 B). This results in a NOT FINISHED message being sent.

An end of SSDU and end of quarantine delimiter is implicit in a FINISH message.

3.2.3.2 The DISCONNECT message

Function

The DISCONNECT message (DN) is used to effect orderly termination of a session connection, in response to a FINISH message. No confirmation is sought as a result of the DISCONNECT message. The disconnection is an implicit end of the current DIALOG-UNIT.

Content

The message contains a code specifying transport disconnection and may contain transparent ss-user data (up to 32 octets).

Sending

A valid "SPM service RELEASE affirmative response" event (EVE 26A), results in a DISCONNECT message. The message is sent on the transport normal flow. Status information relating to the connection is cleared (i.e. the connection ceases to exist). The SPM goes to the state "unconnected" (STA 1).

NOTE 23

There may be an implementation dependent intermediate state for statistics gathering etc. which is not externally distinct from STA 1.

Receiving

A valid "incoming DISCONNECT message" event (EVE 14) normally results in an "SPM service RELEASE affirmative confirmation" event (EVE 116A). The receiving session entity does not respond. Status information relating to the connection is cleared (i.e. the connection ceases to exist). The SPM goes to the state "unconnected" (STA 1) and invokes the transport disconnect service.

NOTE 24

There may be an implementation dependent intermediate state for statistics gathering etc. which is not externally distinct from STA 1.

3.2.3.3 NOT FINISHED message

Function

The NOT FINISHED message (NF) is used to report that the receiver of a FINISH message is not ready to terminate the session-connection. This message can only be sent if the TERMINATE TOKEN is defined.

Content

The message may contain transparent user data (up to 32 octets).

Sending

A valid "SPM service RELEASE negative response" event (EVE 26B) results in a NOT FINISHED message. The message is sent on the transport normal flow. The SPM goes to the state STA 7 or STA 13 according to the current position of the DATA TOKEN.

Receiving

A valid "incoming NOT FINISHED message" event (EVE 33) normally results in an "SPM service RELEASE negative confirmation" event (EVE 116B). The SPM goes to the state STA 7 or STA 13 according to the current position of the DATA TOKEN.

3.2.3.4 The ABORT message

Function

The ABORT message (AB) is used to effect abnormal termination of a session-connection at any time. This includes the establishment, the data transfer, and disconnection phases. This message is also used by a session entity to terminate the session connection when a protocol error is detected. The session

ABORT is an implicit end of the current D-U. The use of this message may cause loss of data.

Content

The message contains either a reason code (2 octets) if it has been issued by the session layer, or transparent ss-user data (3 octets) if it has been generated by the session user. The message contains a code specifying transport disconnection.

Sending

A valid "SPM service ABORT request" event (EVE 2), or the detection of a protocol error in any state of the SPM results in an ABORT message. The message is sent on the transport expedited flow. The SPM goes to the state "unconnected" (STA 1). Status information relating to the connection is cleared (i.e. the connection ceases to exist).

NOTE 25

There may be an implementation-dependent intermediate state for error analysis etc. which is not externally distinct from STA 1.

Receiving

A valid "incoming ABORT message" event (EVE 31) normally results in an "SPM service ABORT indication" event (EVE 102). The receiving session entity shall not transmit on either of the two flows. The receiving session entity does not respond. Status information relating to the connection is cleared (i.e. the connection ceases to exist). The SPM goes to the state "unconnected" STA 1.

NOTE 26

There may be an implementation-dependent intermediate state for error analysis etc. which is not externally distinct from STA 1.

3.2.4 Data Transfer Protocol

Session data transfer protocol carries and delimits session user data. It uses the DATA TRANSFER (DT), MARK CONFIRMATION (MC) and EXPEDITED (EX) messages.

Suring the data transfer phase, the exchange of data (SSDUs) and control informations is regulated by the use of TOKENS (see 1.5.10).

NOTE 27

SS-user data may also be transferred in the CONNECT, ACCEPT, REFUSE, FINISH, ABORT, PLEASE-TOKENS, RESYNCHRONIZE and RESYNCHRONIZE ACK messages.

3.2.4.1 The DATA TRANSFER message

Function

The DATA TRANSFER message (DT) is used to carry session user data and delimiting information between session connection users. It is also used to give tokens. If the data token has been defined for that session connection, only the owner of this token may issue DT. If the synchronize token has been defined for that session connection, only the owner of this token may issue marks types B or C or D. If the end-DU token has been defined, only the owner of this token may issue MARKS type D. The mark type D is used to close normally the current DIALOG-UNIT.

If a mark is indicated, the delimiter type must be end of quarantine of quarantining is on. Marks are used to identify a point in the dialogue. The mark identifier is a serial number which is contained in a variable called V(M). The variable V(M) is incremented by one, by both session entities whenever a mark is sent or received (see 3.2.4.1 Sending and Receiving).

Content

The message may contain:

- user data (SSDU)
- delimiter
- GIVE TOKENS
- mark type and transparent user data (up to 6 octets)

The user data and delimiters may be absent. These are processed in the sequence defined in appendix D.2. The delimiter types are end of fragment, end of SSDU and end of quarantine. If surrender of data token is indicated the delimiter type must be end of quarantine, if quarantining is on.

Marks may be of three types:

- Type B This mark supports the service of minor synchronization point (normal type). A MARK CONFIRMATION (MC) may be returned by the SPM user, but is not required by the session protocol. This mark has no implications on the data flows.
- Type C This mark supports the service of minor synchornization point (urgent type). A MARK CONFIRMATION (MC) is required before all subsequent transmissions on the normal flow. There is no other implication on the data flows.
- Type D This mark supports the service of major synchronization point. A MARK CONFIRMATION (MC) is required before all subsequent data transmission on the normal and expedited flows. The SPM issung this mark will not send a RESYNCHRONIZE request with the op-

tion "restart" until the receipt of the MARK confirmation. This mark is used for closing the current DIALOG-UNIT. The separation between two consecutive D-U is achieved at protocol level by sending a PREPARE message (MARK CONFIRMATION), on the transport expedited flow (see 3.2.4.2 Additional Information). This mark also identifies a point in the dialogue before which resynchronization with the option RESTART is not permitted by the session layer.

Sending

A valid "SPM service TRANSFER request" event (EVE 4), results in a DATA TRANSFER message. The message is sent on the transport normal flow. After sending the message the value of V(M) is incremented by one if any mark type B or C or D was requested. Unless a type C or D mark has been requested, the SPM goes to the state "idle DATA TOKEN not my side" (STA 13), or "idle DATA TOKEN my side" (STA 7), depending on the new position of the DATA TOKEN. If a type C mark has been requested, the SPM goes to the state "waiting for mark C confirmation" STA 12. If a type D mark has been requested, the SPM goes to the state "waiting for MARK D CONFIRMATION message, DATA TOKEN my side" (STA 4A), or "waiting for MARK D CONFIRMATION message, DATA TOKEN not my side" (STA 4B), depending on the new position of the data token.

Receiving

A valid "incoming DATA TRANSFER message" event (EVE 15) normally results in an "SPM service TRANSFER indication" event (EVE 104). If the message carries a mark of type B or C or D, the value of V(M) is incremented by one. Unless the incoming DATA TRANSFER message indicates a type D mark, the SPM goes to the state "idle DATA TOKEN not my side" (STA 13), or "idle DATA TOKEN" (STA 7), depending on the new position of the data token If a type D mark has been indicated, then the SPM goes to the state "waiting for MARK D response DATA TOKEN my side" (STA 10A), or "waiting for MARK D reponse, DATA TOKEN not my side" (STA 10B), depending on the new position of the data token

3.2.4.2 The MARK CONFIRMATION message

Function

the MARK CONFIRMATION message (MC) is used to return a confirmation to a received mark of any type with a session user specified mark serial number. This means that the confirmation refers to a mark serial number and not to a mark type. The confirmation of a mark serial number implies the confirmation of all the previous mark numbers. The confirmations shall be sent in

the increasing order of the serial numbers to be confirmed. As a consequence, the confirmation of a serial number which has already been confirmed, is detected as a protocol error.

Content

The message contains a user specified serial number and transparent ss-user data (up to 6 octets).

Sending

A valid "SPM service MARK response" event (EVE 27), results in a MARK CONFIRMATION message. The message is sent on the transport normal flow. The SPM goes to the state "idle DATA TOKEN not my side" (STA 13), or "idle DATA TOKEN my side" (STA 7), depending on the current position of the data token.

Receiving

A valid "incoming MARK CONFIRMATION message" event (EVE 16) normally results in an "SPM service MARK confirmation" event (EVE 113). The SPM goes to the state "idle DATA TOKEN my side" (STA 7) or "idle DATA TOKEN not my side" (STA 13), depending on the current position of the data token.

Additional information

When the serial number, which has to be confirmed, has been indicated by a type D mark, a PREPARE message shall be sent immediately before the MARK CONFIRMATION in order to isolate the current DIALOG-UNIT from the next one (normal termination).
The last confirmation sent (serial number a) and the mark received (serial number b) define an interval:
/ a, b / with b greater or equal to a
a does not belong to the interval
b belongs to the interval.

All confirmation messages shall identify a point within the interval / a , b / . Otherwise a protocol error is signalled.

NOTE 28

After sending a confirmation message, related to a serial number identified by a type C or D mark, the above interval becomes empty.

NOTE 29

The ability to send mark confirmations is only on the side that receives the marks.

3.2.4.3 The EXPEDITED message

Function

The EXPEDITED message (EX) is used to transfer small amounts (no more than 6 octets) of ss-user data asyn-

chronously and independent of the TOKENS and independent of the enclosures of SSDUs in DATA TRANSFER messages.

Content

The message carries a limited amount of user data (6 octets).

Sending

A valid "SPM service EXPEDITED request" event (EVE 6) results in an EXPEDITED message. The message is sent on the transport expedited flow. The SPM state is not changed.

Receiving

A valid "incoming EXPEDITED message" event (EVE 18) normally results in an "SPM service EXPEDITED indication" event (EVE 106). The SPM state is not changed.

3.2.5 Synchronization Protocol

Session connection synchronization protocol is used to synchronize the exchange of data between the two users of the session connection, typically in error and exception circumstances. It may be used to close a DIALOG-UNIT. This protocol uses the CANCEL (CL), RESYNCHRONIZE (RS), RESYNCHRONIZE ACKNOWLEDGEMENT (RA), PLEASE TOKENS (PT), GIVE TOKENS (GT) and PREPARE (PR) messages.

3.2.5.1 The CANCEL message

Function

The CANCEL message (CL) is used to cancel data which has not yet been delimited by end of quarantine.

Content

There is no parameter.

Sending

A valid "SPM service CANCEL request" event (EVE 5) results in a CANCEL (CL) message. The message is sent on the transport normal flow. The SPM state is not changed.

Receiving

A valid "incoming CANCEL message" event (EVE 17) results in the receiving session entity discarding all the SSDUs belonging to the current quarantine unit. The SPM state is not changed.

Additional information

The receipt of this message may not be visible to the session service user, because SSDUs are not delivered until after the current quarantine unit is terminated.

3.2.5.2 The RESYNCHRONIZE message

Function

The RESYNCHRONIZE message (RS) is used to provide to the ss-user a selective means to resynchronize the exchange of data to a previous mark and to reposition the tokens to an agreed side. Typically, this protocol is used after failure, and may imply additional destruction of data.

NOTE 30

The use of the resynchronize service may cause destruction of data (see 3.2.5.6) but the message RS itself does not destroy anything.

This protocol can also be used to "purge" or "reset" the connection, since that is a particular case of resynchronization.

The RESYNCHRONIZE message may be used to abnormally close the current DIALOG-UNIT. Two options are provided:

- abandon (the current unit of work is discarded)
- restart (the current unit of work will be restarted in a new D-U).

NOTE 31

These options are user provided semantics and are not seen by the session layer except for solving contention cases.

Since the protocol provides a repositioning of the tokens a particular use of it is the destructive way to get tokens.

Content

The message contains:

- user specified mark serial number (see note 32)
- user specified tokens positions (initiator, acceptor, acceptor choses)
- option: abandon/restart
- transparent user data (up to 6 octets) optional

NOTE 32

When the abandon option is invoked, the SPM supplies the current value of V(M) as mark serial number.

Sending

A valid "SPM service RESYNCHRONIZE request" event (EVE 7) results in a RESYNCHRONIZE message which is sent on the transport normal flow. The SPM goes to the state "waiting for RESYNCHRONIZE ACKNOWLEDGEMENT message" (STA 5). In this state, all the incoming messages are discarded except RESYNCHRONIZE (see Additional information), RESYNCHRONIZE ACK and ABORT.

Receiving

A valid "incoming RESYNCHRONIZE message" event (EVE 19) normally results in an "SPM service RESYNCHRONIZE indication" event (EVE 107). The SPM goes to the state "waiting for RESYNCHRONIZE response event" (STA 11).

Additional Information

A PREPARE message shall also be sent just before the RESYNCHRONIZE message.
The contention between two RESYNCHRONIZE messages is solved according to the following table:

		Incoming RESYNCHRONIZE message from SPM B	
		option = ABANDON	option = RESTART
outgoing RESYNCHRONIZE message from SPM A	option = ABANDON	contention winner (result:ABANDON)	SPM A's request (result:ABANDON)
	option = RESTART	SPM B's request (result: ABANDON)	. lowest mark serial number or . contention winner if equality (result: RESTART)

The table describes how SPM A shall behave. An equivalent behaviour could be deduced for SPM B. The general rule is: only one of the two requests is taken into account; the other is discarded.

Explanations:

- contention winner: only the message, issued by the contention winner (see 3.2.2.1 Additional Information) is taken into account.
- SPM X's request: only the request, isseud by the SPM X is taken into account.
- Lowest mark serial number: the lowest mark serial number identifies the request which is taken into account.

If an incoming RESYNCHRONIZE message (with the restart option) is not acceptable, the receiving ss-user may issue another request (EVE 7) with a lower serial number or with the abandon option. In general, a counter-proposal may be issued if it prevails over the original proposal according to the above decision rules.

3.2.5.3 The RESYNCHRONIZE ACKNOWLEDGEMENT message

Function

The RESYNCHRONIZE ACKNOWLEDGEMENT message (RA) is used to notify to the sender of a RESYNCHRONIZE message the completion of re-synchronization.

Content

The message contains:

- a mark serial number
- user specified TOKEN positions (initiator, acceptor) which must be consistent with those proposed in the incoming request.

- transparent ss-user data (up to 6 octets), optional.

When the "abandon" option is used, the SPM supplies the current value of V(M) as mark serial number. When the "restart" option is used, the ss-user supplies the mark serial number.

Sending

A valid "SPM service RESYNCHRONIZE response" event (EVE 28) results in a RESYNCHRONIZE ACKNOWLEDGEMENT message. The message is sent on the transport normal flow. The SPM goes to the state "idle DATA TOKEN my side" (STA 7) or "idle DATA TOKEN not my side" (STA 13), depending in the new position of the data token.. The value of V(M) is set:

- to the mark serial number agreed as the resynchronization point, for the option restart,
- to the higher serial number contained in the RS and RA messages, for the option abandon.

Receiving

A valid "incoming RESYNCHRONIZE ACKNOWLEDGEMENT message" event (EVE 20) normally results in an "SPM service RESYNCHRONIZE confirmation" event (EVE 114). The tokens are set to the positions indicated. The SPM goes to the state "idle DATA TOKEN my side" (STA 7) or "idle DATA TOKEN not my side (STA 13), depending on the new position of the data token. The value of V(M) is set:

- to the mark serial number agreed as the resynchronization point, for the option restart,
- to the higher serial number contained in the RS and RA messages, for the option abandon.

Additional information

A PREPARE message shall also be sent just before the RESYNCHRONIZE ACKNOWLEDGEMENT message (see 2.3.5.6, Sending).

The result of the exchange of the two messages (RS and RA) is that the two users of the session connection have negotiated who has what TOKEN and the identity of the mark from which they will resume their exchange of data.

3.2.5.4 The PLEASE TOKENS message

Function

The PLEASE TOKENS message (PT) is used to ask for one or several tokens in a non-destructive way.

Content

The message contains:

- types of the requested tokens
- transparent ss-user data (up to 1 octet) optional.

Sending

A valid "SPM service PLEASE TOKENS request" event (EVE 9) results in a PLEASE TOKENS message. The message is sent on the transport normal flow. The SPM state is not changed.

Receiving

A valid "incoming PLEASE TOKENS message" event (EVE 23) normally results in an "SPM service PLEASE TOKENS indication" event (EVE 109). The SPM state is not changed.

Additional Information

If the receiving ss-user has the requested token, it should surrender it promptly, but need not do so at all. The user surrenders TOKENS by the usual means, i.e. a DATA TRANSFER message, with or without data, or a GIVE TOKENS message.

3.2.5.5. The GIVE TOKENS message

Function

The GIVE TOKENS message (GT) is used to give to the partner, one or several tokens among: Data, Synchronize, End-DU and Terminate tokens. Only the owner of a token is allowed to give it through this message.

Content

The message contains the tokens which are given.

Sending

A valid "SPM service GIVE TOKENS request" event (EVE 8) results in a GIVE TOKENS message. The message is sent on the transport normal flow. The SPM state is changed according to the new positions of the tokens.

Receiving

A valid "incoming GIVE TOKENS message" event (EVE 21) normally results in an "SPM service GIVE TOKENS indica-

tion" event (EVE 108). The SPM state is changed according to the new positions of the tokens.

3.2.5.6 The PREPARE message

Function

The PREPARE (PR) message is used to notify the imminent arrival of certain message types which are sent unexpectedly. Its effects may be destructive.

Content

The message contains a parameter value identifying the imminent message type.

Sending

The PREPARE message is sent before the following messages: RESYNCHRONIZE, RESYNCHRONIZE ACKNOWLEDGEMENT, and MARK CONFIRMATION type D.
The PREPARE message is sent on the transport expedited flow (the other message is sent on the transport normal flow). The SPM goes to the state which is determined by the initial request (see 3.2.5.2, Sending; 3.2.5.3 Sending; 3.2.4.2, Sending).

Receiving

A valid "incoming PREPARE message" event (EVE 32) normally results in the state "waiting after PREPARE message" (STA 15). There is no corresponding SPM service indication event.

In this state, incoming messages on the transport normal flow are dealt according to rules specific to the message type indicated by the parameter value of the PREPARE message (see 3.2.5.2 Receiving; 3.2.5.3. Receiving; 3.2.4.2 Receiving).

3.3 TRANSPORT-SERVICE-MAPPING PROTOCOL

3.3.1 General

This clause defines the transport service mapping protocol (TSM protocol), including definition of the service required from the transport layer. One transport mapping is defined; it has no variants and no options.

3.3.2 Transport service

The transport services which are used in this protocol are defined in Standard ECMA-72.

3.3.3 Connection mapping

The connection-oriented type of transport service is used and no other.
The transport connection shallhave the expedited data transfer capability.
The transport-connection may have the purge capability, but this is not used.

Each session-connection uses one transport-connection. The transport-connection is used by one session-connection only; it is not used in any other way.

3.3.4 Transport Connection Establishment

The transport connection is established before any SPDU protocol activity (see note 33).
The transport connection establishment is initiated by the session entity which initiates establishment of the session connection (see notes 34 and 35).
The data parameters of the transport connection establishment phase are not used. The use of the parameters relating to the transport expedited and purge options is defined in 3.3.3. All other parameters of the transport connection establishment phase are handled by functions within the session layer, in ways undefined by this standard.

NOTE 33

Typically the transport connection is established immediately before the session connection.

NOTE 34

The assymmetry avoids the possibility of collision between session connection requests on the same transport connection.

NOTE 35

Multiple session connection establishment requests are not generally visible to the session service and protocol. The collision of two session connection establishment requests on the same transport connection is a special case. Because there is no multiplexing, this is necessarily visible to, and avoided or resolved by, the session protocol.

3.3.5 Transport Connection Data Phase

All of the messages of the SPDU protocol are mapped into the Data Transfer and Expedited Data Transfer primitives of Standard ECMA-72. SPDU protocol is not mapped into any other transport service primitives. The mapping is one-to-one (see note 36). The transport normal flow is subject to back pressure flow control by functions within the session layer, in ways undefined by this standard. The transport expedited flow should generally be used in such a way that it is never blocked by back pressure (see note 37). The transport expedited flow shall not be blocked by back pressure unless the transport normal flow is also blocked by back pressure.

NOTE 36

The size of each SPDU is delimited by end of TSDU, and its size is therefore necessarily a multiple of 8 bits.

NOTE 37

The transport expedited flow is systematically used by the SPM protocol as a way of avoiding potential transport flow control

deadlocks. SPM message types which may be sent unexpectedly are generally sent on the transport expedited flow.

3.3.6 Transport Connection Termination

The transport connection termination is initiated in one of two ways:

- by an SPM acting with prior agreement of the other SPM; or
- spontaneously by the transport service.

The first case arises only after completion of SPDU protocol which terminates the session connection. One SPM receives a REFUSE, DISCONNECT or ABORT message with the transport disconnection code "terminate". This SPM then issues a disconnection request to the transport service.

The second case may arise at any time after initiating transport connection establishment, including during the first case or during the SPDU protocol leading up to it. It is treated as a transport service failure, and results in automatic termination of the session connection (see note 38) with the appropriate reason code, i.e. S-ABORT indication.

NOTE 38

Spontaneous transport connection termination (i.e. transport failure) after the SPM has initiated transport termination may result in the other SPM being unable to know whether the SPDU protocol completed orderly termination of the session connection.

3.4 PROTOCOL ENCODING

3.4.1 General Principles

Each SPDU is mapped into a Transport Service Data Unit (TSDU) one-to-one, as specified in 3.3. Each SPDU is therefore delimited by the end of a TSDU, and is a multiple of 8 bits.

The messages defined in 3.2 are generally encoded one-to-one into SPDUs. The only exception is the DATA TRANSFER message (see 3.2.4.1) whose encoding may span an arbitrary (even fractional) number of SPDUs if user data are segmented or blocked (see 3.4.3).

An SPDU is encoded with two distinct parts.

- Fixed Header (FH)
- Variable Information Unit (VIU)

This general structure is illustrated in figure 3.4/1

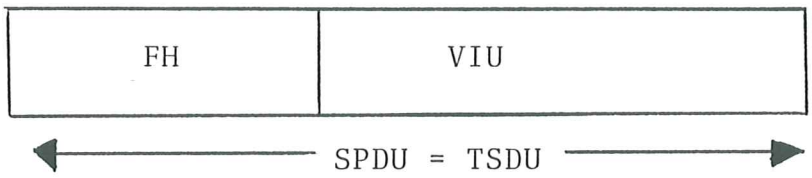


Fig. 3.4/1 - SPDU General Structure

The fixed header encodes those characteristics which are most frequently used or which most fundamentally affect the structure of the protocol. It also defines the structure of the following variable information unit.

The variable information unit contains items encoding the session protocol information, and items with transparently-carried user data; both kinds of information can appear in various combinations.

Within the variable information unit, items with session protocol information are "TLV-encoded" in a format described in 3.4.2 which includes an item "type" information. Items with user data may also be TLV-encoded, or may not have any structure visible to the session protocol.

TLV-encoded items (TLV items) containing protocol information are distinguished from TLV items containing user data by their "type" field. A set of types is reserved for session protocol TLV items (see 3.4.5) other types are available to identify TLV items for ss-user data in SSDU blocking (see 3.4.3).

The encoding is designed to allow future extensions; there are reserved encodings in the fixed header, and the information unit encoding is inherently flexible and extendable.

All the fields which are qualified as "reserved" shall contain a bit configuration of all ZERO.

3.4.2 Encoding of TLV items

Each TLV item carries three distinct pieces of information (explicitly or implicitly).

- a type
- a length
- a value

These pieces of information are contained in adjacent fields of the TLV item.

The formats which are defined below are classified as:

F1 : implicit length, no value field
F2 : implicit length, 1-octet value field
F3 : explicit length, n-octet value field

Each format (F1, F2, F3) will be defined field by field.

First field of formats F1, F2 and F3.

This field contains the type information.
1 octet (b1, b2 ...b8).
b1 = 1 : fixed length formats

b2 = 0 - no value field (format F1)

b3 to b8 = 0 - reserved
≠ 0 - format F1 types
(range 129-191 decimal)

b2 = 1 - 1-octet value field follows (format F2)

b3 to b8 = 0 - reserved
≠ 0 - format F2 types
(range 193-255 decimal)

b1 = 0 - variable length value field (format F3)

b2 to b8 = 0 - reserved
≠ 0 - format F3 types
(range 1-127 decimal)

Second field of format F2

This field contains the value information.
1 octet (b1, b2 ...b8).
b1 to b8 - 1-octet data

Second field of format F3

This field contains the length information.
1-octet (b1, b2 ...b8) or 2-octets (b1, b2 ...b16)

b1 = 0 - the length is specified in the 7 bits which follow (b2 to b8) (range 0-127)
b1 = 1 - the length is specified in the 15 bits which follow (b2 to b16) (range 0-32767)

b2 to b8 or b2 to b16 - length (in octets) of the value field.

Third field of format F3.

This field contains the value information:
value (of the length specified in the second field).

3.4.3 SSDU blocking TLV items

For SSDU blocking, each SSDU is encoded as a TLV item with a type 16 or outside the range reserved for session protocol types (see 3.4.5).

SSDU blocking TLV items shall be positioned in the order in which they were submitted to the session service.

An SSDU blocking TLV may span across two or more SPDUs. This characteristic is not explicitly encoded. The spanning characteristics are implicit because the TLV in question has a length which makes it continue beyond end of SPDU. Its next octet is where SSDU blocking TLVs start in the next SPDU.

To simplify the decoding, it is stated that the type and the length of this user item shall not be split in two SPDUs.

NOTE 39

To implement this SSDU blocking, the session layer inspects each SSDU submitted for blocking. If it is already in a correct TLV format with a non reserved type (see 3.4.5) then no further encoding of it is necessary. Otherwise the SSDU is encoded as a TLV of type 16 while it is inside the session layer and this encoding is stripped off when it leaves the session layer. The non reserved types are available for use by data structuring protocols in presentation layer. The SSDU blocking capability of session layer is independent of whether presentation layer actually uses TLV encoding and the non reserved types, although that gives the optimum result.

3.4.4 Fixed Header

The SPDU fixed header contains information on:

- the category of the SPDU itself (use described in 3, 4, 5);
- the structure of the SPDU information unit (structures defined in 3.4.5);
- "enclosures" which delimit user data, if any;
- type of a mark, if present.

The size of the fixed header part of an SPDU is two octets. The format is defined in figure 3.4/2. The encoding is defined in table 3.4/3.

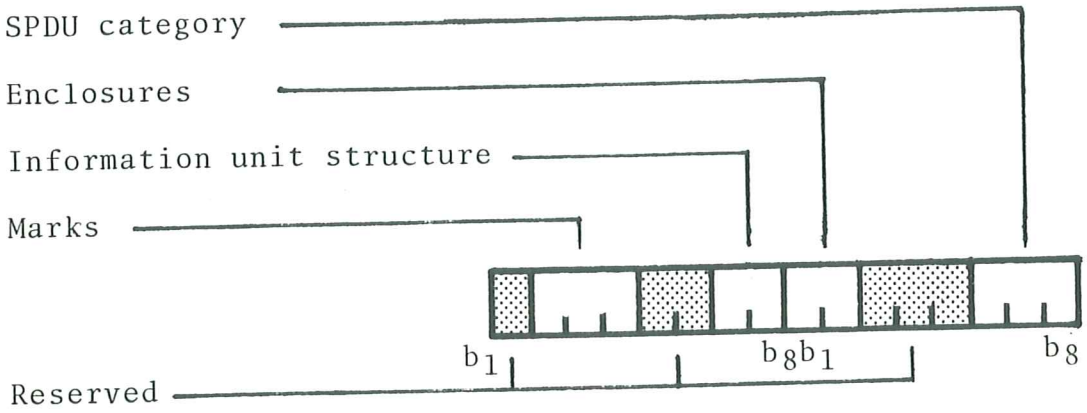


Fig. 3.4/2 - Format of Fixed Header

Table 3.4/3 - Encoding Fixed Header

FIELD	VALUE	MEANING
SPDU category (see 3.4.5)	0 1 2 3 4 5 6 7	DATA category EXPEDITED category NORMAL FLOW category CONNECTION category DISCONNECTION category SYNCHRONIZATION category reserved reserved
Enclosures (see 3.4.6.1)	0 1 2 3	No enclosures (none) End of quarantine unit (EOQ) Give DATA token (GDT) and EOQ if applicable reserved
Information unit structure (see 3.4.5)	0 1 2 3	Structure 0 Structure 1 Structure 2 Structure 3
Marks (see 3.4.6.1)	0 1 2 3 4 to 7	No marks Mark type B (B-mark) Mark type C (C-mark) Mark type D (D-mark) Reserved

3.4.5 Variable information unit

The contents and possible structures of the SPDU variable information unit depend on the message which is encoded in the SPDU. The category of the SPDUs carrying the information for each message is defined in table 3.4/4. The SPDU category is indicated in the fixed header (see 3.4.4).

Table 3.4/4 - messages/SPDU category correspondance

Message Name	Category
DATA TRANSFER	DATA
EXPEDITED	EXPEDITED
CANCEL MARK CONFIRMATION GIVE TOKENS PLEASE TOKENS	NORMAL FLOW
CONNECT ACCEPT REFUSE	CONNECTION
FINISH DISCONNECT NOT FINISHED ABORT	DISCONNECTION
RESYNCHRONIZE RESYNCHRONIZE ACK PREPARE	SYNCHRONIZATION

There are four possible structures of the variable information unit, as defined in fig. 3.4/5. The structure is indicated in the fixed header (see 3.4.4)

Structure 0

(last segment)

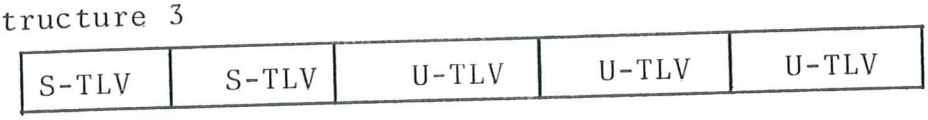
Structure 1

(not last segment)

Structure 2

(last segment)

Structure 3



where S-TLV: session protocol TLV item
U-TLV: session user data TLV item
TRANSPARENT: transparent session user data

Fig. 3.4/5 - Variable Information Unit Structures

The number and type of the required S-TLV items is determined in each SPDU which has structure 2 or 3; the order in which they must appear is also determined by the session protocol. The U-TLV items, when present, shall follow all S-TLV items. Only one (possibly empty) transparent user data field may appear in structures 0, 1 and 2, delimited by the end of the SPDU/TSDU. Any number of user TLV items may appear in structure 3. Structures 0 and 1 only differ for the information they carry about the segmentation of user data. The complete list of the TLV items used in protocol messages is given in table 3.4/6. The table also indicates the types used in the S-TLVs. For each item, a format (f1, F2, F3) and a type (depending on the format) are indicated.
Subsection 3.4 specifies the mandatory items and the optional items in the SPDU variable information unit for each message. It also defines the contents of the value field of each protocol TLV-ed item.

Table 3.4/6 - Items defined in each SPDU category

SPDU category	Item	TLV format	Type (see note 40)
DATA	Session-TLVed User Data	F3 (see note 41)	16 (see note 41)
	Mark/Token Extension	F3	17
EXPEDITED			
NORMAL FLOW	Cancel	F1	129
	Mark Confirmation	F3	18
	Give Tokens	F2	207
	Please Tokens	F2	206
CONNECTION	Protocol Identification	F2	205
	Connect	F3	19
	Accept	F3	20
	Refuse	F3	21
	Initiator Address	F3	22
	Acceptor Address	F3	23
	Connection Identifier	F3	24
DISCONNECTION	Finish	F2	204
	Disconnect	F2	203
	Not finished	F1	130
	Abort	F3	25
SYNCHRONIZATION	Resync	F3	26
	Resync Ack	F3	27
	Prepare	F2	202

NOTE 40

The type is given as the decimal representation of the first field of the TLV item, which also indicates the format (see 3.4.2)

NOTE 41

Any other type not reserved for the session protocol is acceptable for user TLV items, when the item is provided directly by the user.

Other types are reserved for future protocol extensions and are not available to the users; the complete set of protocol and user types are given in table 3.4/7.

Table 3.4/7 - Allocation of Types

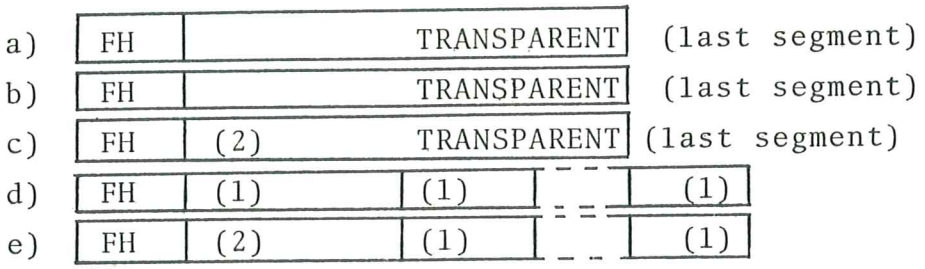
Format	Session Protocol Types	User Types
F1	129 + 143	144 + 191
F2	193 + 207	208 + 255
F3	1 + 31	32 + 127

3.4.6 SPDU Encoding of Protocol Messages

This subsection defines, for each message, the format and content of the SPDU(s) encoding it. The internal encoding of protocol TLV items is defined in the notation of PL/1 structure declarations.

3.4.6.1 DATA TRANSFER Message

Encodings



(1) - Session TLVed User Data Item
(2) - Mark/Token Extension item

Fig. 3.4/8.

Fixed Header

Table 3.4/9

Encoding	SPDU category	Enclosures	Information structure	Marks
a)	DATA	None, EOQ or GDT	0	Any
b)	DATA	None	1	None
c)	DATA	None, EOQ or GDT	2	Any
d)	DATA	None, EOQ or GDT (see note 43)	3	Any (see note 44)
e)	DATA	None, EOQ or GDT (see note 43)	3	Any (see note 44)

NOTE 42

Encodings d) and e) are used for blocking SSDUs.

NOTE 43

With encodings d) and e) when the last ss-user TLV spans across the SPDU boundary, the Enclosure can only be None and the Marks field can only be No marks.

NOTE 44

Restrictions defined in Appendix D.2 limit the acceptable combinations of marks and tokens.

Information Unit

In encodings a), b) and c) the user transparent data may be omitted.

- Item 1: Session TLVed User Data Item

  Format: F3
  type: 16
  length: variable
  value: transparent to session layer

  See also note 41, clause 3.4.5, table 3.4/6.

- Item 2: Mark/Token Extension item

  Format: F3
  type: 17
  length: variable from 2 to 8
  value: declare 1 Mark/Token Extension item

```
2 Reserved              bit (2)
2 SYNCHRONIZE token     bit (2)
2 TERMINATE token       bit (2)
2 End-DU token          bit (2)
2 Reserved              bit (8)
2 ss-user  mark data    bit (48) (maximum)

The 2-bit field values for each token:
0 no action
1 give token
2 reserved
3 reserved
```

NOTE 45

The ss-user data field can vary from 0 to 6 octets (see 2.2.2.9, 2.2.2.10).

3.4.6.2 EXPEDITED message

Encoding

FH	TRANSPARENT

Fig. 3.4/10

Fixed Header

Table 3.4/11

SPDU category	Enclosures	Information Structure	Marks
EXPEDITED	None	0	No marks

Information Unit

The information unit length ranges from 0 to 6 octets.

3.4.6.3 CANCEL message

Encoding

FH	(1)

(1) - Cancel item

Fig. 3.4/12

Fixed Header

Table 3.4/13

SPDU category	Enclosures	Information Structure	Marks
NORMAL FLOW	None	2	No marks

Information Unit

The user transparent data are not permitted

- Item 1: CANCEL item

  format: F1
  type: 129

3.4.6.4 MARK CONFIRMATION message

Encoding

FH	(1)

(1) - Mark Confirmation item

Fig. 3.4/14

Fixed Header

See table 3.4/13

Information Unit

The user transparent data are not permitted.

- Item 1: Mark Confirmation item

format: F3
type: 18
length: variable from 2 to 8
value: declare 1 Mark Confirmation item
2 mark serial number bit (16)
2 ss-user data bit (48) (maximum)

NOTE 46

The ss-user data field can vary from 0 to 6 octets (2.2.2.9, 2.2.2.10).

3.4.6.5 GIVE TOKENS message

Encoding

FH	(1)

(1) - Give tokens item

Fig. 3.4/15

Fixed Header

See table 3.4/13

Information Unit

The user transparent data are not permitted.

- Item 1: GIVE TOKENS item

format: F2
type: 207
value: declare 1 Give Tokens item
2 DATA token bit (1)
2 SYNCHRONIZE token bit (1)
2 TERMINATE token bit (1)
2 END D-U token bit (1)
2 Reserved bit (4)

for each token: 0: token not given
1: token given

3.4.6.6 PLEASE TOKENS message

Encoding

FH	(1)	TRANSPARENT

(1) - Please tokens item

Fig. 3.4/16

Fixed Header

See table 3.4/13

Information Unit

The user transparent data may be absent (size is 1 octet); the TLV item is mandatory.

- Item 1: - Please tokens item

format: F2
type: 206
value: declare 1 Please Tokens item like Give tokens item

for each token: 0: token not asked for
1: token asked for

3.4.6.7 CONNECT message

Encoding

FH	(1)	(2)	(3)	(4)	(5)	TRANSPARENT

(1) - Protocol Identification item
(2) - Connect item
(3) - Initiator Address item
(4) - Acceptor Address item
(5) - Connection Identifier item

Fig. 3.4/17

Fixed Header

Table 3.4/18

SPDU Category	Enclosures	Information structure	Marks
CONNECTION	None	2	No marks

Information Unit

The user transparent data may be absent (maximum size is 64 octets); the TLV items are mandatory.

- Item 1: - Protocol Identification item

format: F2
type: 205
value: declare 1 Protocol Identification item
2 ECMA protocol version bit (4) (see 1)
2 protocol subset bit (4) (see 2)

1 - value is 1; all other values are reserved
2 - 1: subset A
2: subset B
3: subset C
4: subset D
all other values are reserved

- Item 2: - Connect item

format: F3
type: 19
length: 19
value: declare 1 Connect item
2 use of tokens (see 1)
3 DATA token bit (2)
3 SYNCHRONIZE token bit (2)
3 END D-U token bit (2)
3 TERMINATE token bit (2)
3 reserved

```
        2 token setting (see 2)
            3 DATA token                bit (2)
            3 SYNCHRONIZE token         bit (2)
            3 END D-U token             bit (2)
            3 TERMINATE token           bit (2)
            3 reserved                  bit (8)

        2 quarantining max size (see 3)
            3 initiator to acceptor bit (16)
            3 acceptor to initiator bit (16)

        2 SPDU max size (see 4)
            3 initiator to acceptor bit (16)
            3 acceptor to initiator bit (16)

        2 SSDU max size (see 5)
            3 initiator to acceptor bit (16)
            3 acceptor to initiator bit (16)

        2 initial mark serial number (see 6) bit (16)

        2 blocking (see 7)
            3 initiator to acceptor bit (1)
            3 acceptor to initiator bit (1)
            3 reserved                  bit (6)
```

1 - use of token (2-bit field):
    0 - token not defined
    1 - token defined
    2 - reserved
    3 - reserved

2 - token setting (2-bit field):
    0 - initiator
    1 - acceptor
    2 - acceptor chooses
    3 - reserved

3 - quarantining capacity (16-bit field):
    0 - quarantining not required
    16-bit unsigned integer: gives maximum quarantining unit size divided by 128

4 - SPDU size:
    16-bit unsigned integer; minimum value: 36

5 - SSDU size:
    16-bit unsigned integer

6 - initial mark serial number
    16-bit unsigned integer

7 - blocking (1-bit field):
    0 - no
    1 - yes

- Item 3: - Initiator Address item

  format: F3
  type: 22
  length: variable (up to 16)
  value: not defined

- Item 4: - Acceptor Address item

  format: F3
  type: 23
  length: variable (up to 16)
  value: not defined

- Item 5: - Connection Identifier item

  format: F3
  type: 24
  length: variable (up to 16)
  value: not defined

3.4.6.8 ACCEPT message

Encoding

FH	(1)	(2)	(3)	(4)	(5)	TRANSPARENT

(1) - Protocol Identification item
(2) - Accept item
(3) - Initiator Address item
(4) - Acceptor Address item
(5) - Connection Identifier item

Fig. 3.4/19

Fixed Header

See table 3.4/18

Information unit

The user transparent data may be absent (maximum size 64 octets); the TLV items are mandatory.

TLV items (1), (3), (4), (5) above are defined for the CONNECT message. (see 3.4.6.7, Information unit).

- Item 2: - Accept item

  format: F3
  type: 20
  length: 19
  value: declare 1 Accept item like Connect item
  (see 3.4.6.7, Information unit)

In the 2-bit field "token setting" for each token, value "2" is reserved in the Accept item.

3.4.6.9 REFUSE message

Encoding

FH	(1)	(2)	TRANSPARENT

(1) - Protocol Identification item
(2) - Refuse item

Fig. 3.4/20

Fixed Header

See table 3.4/18

Information Unit

The user transparent data may be absent (maximum size is 64 octets); the TLV items are mandatory.

- Item 1: - Protocol Identification item

  The Protocol Identification item is defined in 3.4.6.7, Information unit, item 1.

- Item 2: - Refuse item

  format: F3
  type: 21
  length: 3
  value: declare 1 Refuse item

2 transport disconnect	bit (1)
2 reserved	bit (7)
2 reason code	bit (16)

  Reason code values:
  0 - reason not specified
  1 - protocol version unknown
  2 - subset not supported
  3 - unacceptable session connection parameters
  4 - acceptor address invalid
  5 - rejection by acceptor ss-user
  all other values are reserved

  Transport disconnect field values:
  0 - reserved
  1 - yes

3.4.6.10 FINISH message

Encoding

FH	(1)	TRANSPARENT

(1) - Finish item

Fig. 3.4/21

Fixed Header

Table 3.4/22

SPDU category	Enclosures	Information structure	Marks
DISCONNECTION	None	2	No marks

Information unit

The user transparent data may be absent (maximum size is 32 octets).

- Item 1: - Finish item

  format: F2
  type: 204
  value: declare 1 Finish item
         2 reserved bit (8)

3.4.6.11 DISCONNECT message

Encoding

FH	(1)	TRANSPARENT

(1) - Disconnect item

Fig. 3.4/23

Fixed Header

See table 3.4/22

Information unit

The user transparent data may be absent (maximum size is 32 octets).

- Item 1: - Disconnect item

  format: F2
  type: 203
  value: declare 1 Disconnect item
         2 transport disconnection bit (1)
         2 reserved bit (7)

         Transport disconnection field values:
         0 - reserved
         1 - yes

3.4.6.12 NOT FINISHED message

Encoding

FH	(1)	TRANSPARENT

(1) - Not Finished item

Fig. 3.4/24

Fixed Header

See table 3.4/22

Information unit

The user transparent data may be absent (max. size is 32 octets).

- Item 1: - Not Finished item

  format: F1
  type: 130

3.4.6.13 ABORT message

Encoding

FH	(1)	TRANSPARENT

(1) - Abort item

Fig. 3.4/25

Fixed Header

See table 3.4/22

Information unit

In an ss-user-generated ABORT message: the Abort item has a length field 1; the ss-user transparent data may be absent (maximum size is 3 octets).

In a service-generated ABORT message: the Abort item has a length of 3 and includes a reason code; the user transparent data are not present.

- Item 1: - Abort item

  format: F3
  type: 25
  length: 1 or 3
  value: declare 1 Abort item

2 transport disconnect	bit (1) (see 1)
2 reserved	bit (7)
2 reason code	bit (16) (see 2)

(only present in service-generated abort)

1 - transport disconnection field values:
  0 - reserved
  1 - yes

2 - reason code field values:
  0 - reason not specified
  6 - protocol error
  all other values are reserved

3.4.6.14 RESYNCHRONIZE message

Encoding

FH	(1)	TRANSPARENT

(1) - Resync item

Fig. 3.4/26

Fixed Header

Table 3.4/27

SPDU category	Enclosures	Information structure	Marks
SYNCHRONIZATION	None	2	No marks

Information unit

The user transparent data may be absent (maximum size is 6 octets).

- Item 1: - Resync item

```
format: F3
type: 26
length: 5
value: declare 1 resync item
       2 mark serial number          bit (16)
       2 token setting
         like Connect item. token setting
         ( see 3.4.6.7, Information unit, Connect item)
       2 resync type                 bit (1)
       2 reserved                    bit (7)

       Resync type field values
       0 - restart
       1 - abandon
```

3.4.6.15 RESYNCHRONIZE ACK message

Encoding

FH	(1)	TRANSPARENT

(1) - Resync Ack item

Fig. 3.4/28

Fixed Header

See table 3.4/27

Information unit

The transparent user data may be absent (maximum size is 6 octets).

- Item 1: - Resync Ack item

```
format: F3
type: 27
length: 5
value: declare 1 Resync Ack item
       like Resync item
```

In the 2-bit field "token setting" for each token value "2" is reserved. Field "resync type" is reserved.

3.4.6.16 PREPARE message

Encoding

FH	(1)

(1) - Prepare item

Fig. 3.4/29

Fixed Header

See table 3.4/27

Information Unit

The user transparent data are not permitted

- Item 1: - Prepare item

format: F2
type: 202
value: declare 1 Prepare item
2 item type bit (8)

Item type (8-bit field) contains the TLV type identifying the message which the PREPARE message anticipates, viz.:
18 - MARK CONFIRMATION
26 - RESYNCHRONIZE
27 - RESYNCHRONIZE ACK
all other values are reserved.

3.5 PROTOCOL SUBSETS

3.5.1 General

For each of the service subsets defined in subsection 2.3, a corresponding protocol subset is defined in this subsection.

3.5.2 Subset A : Basic Subset

The SPDU protocol messages included in this subset are:

- CONNECT
- ACCEPT
- REFUSE
- FINISH
- DISCONNECT
- ABORT
- DATA TRANSFER
- EXPEDITED

Parameters and parameter values wich relate to services and messages that are not included in the subset are not used.

The TSM protocol is defined in subsection 3.3.

3.5.3 Subset B : Basic Interactive Subset

The SPDU protocol messages included in this subset are:

- all the messages included in Subset A
- CANCEL
- PLEASE TOKENS
- GIVE TOKENS

Parameters and parameter values which relate to services and messages that are not included in the subset are not used.

The TSM protocol is defined in subsection 3.3.

3.5.4 Subset C : Basic Synchronized Subset

The SPDU protocol messages included in this subset are:

- all messages included in Subset A, except EXPEDITED
- MARK CONFIRMATION
- RESYNCHRONIZE
- RESYNCHRONIZE ACKNOWLEDGEMENT
- PLEASE TOKENS
- GIVE TOKENS
- PREPARE
- NOT FINISHED

Parameters and parameter values which relate to services and messages that are not included in the subset are not used.

The TSM protocol is defined in subsection 3.3.

3.5.5 Subset D : Basic TWA Subset

The SPDU protocol messages included in this subset are:

- all the messages included in Subset A, except EXPEDITED
- PLEASE TOKENS
- GIVE TOKENS

Parameters and parameter values which relate to services and messages that are not included in the subset are not used.

The TSM protocol is defined in subsection 3.3.

4 Conformance Requirements

4.1 GENERAL

This clause defines the conformance requirements for the session-protocol defined in section 3 of this standard.

There is no conformance requirement for the abstract session-service defined in section 2 of this standard.

4.2 EQUIPMENT

The conformance requirement is for equipment which consists of hardware and/or software and has the purpose of conforming with this standard. The equipment may also have other purposes.

4.3 PEER EQUIPMENT

Any execution of the protocol necessarily involves a peer equipment with which the subject equipment communicates.

For purposes of verifying conformance it is assumed that this other peer equipment:

- is operating in conformance with the standard;
- may be capable of controlled deviation, in that it may be the source of deliberate protocol-errors for the purpose of testing.

This conformance requirement does not distinguish any differences of status between the two equipments (i.e. the notion of a "reference equipment" with a "definitive implementation" is not used).

4.4 PROTOCOL SUBSETS

The equipment shall implement one, or more, of the protocol subsets defined in subsection 3.5.

The supplier of the equipment shall nominate which of these subsets the equipment is intended to conform with.

4.5 ADDITIONAL SESSION PROTOCOL

In addition to the subsets nominated as in 4.4, the equipment may also implement other session layer protocol, including different subsets of the protocol defined in this standard.

Such additional provisions are themselves not in conformance with the standard, but do not prejudice conformance with the standard provided that they are separate and do not prevent use of the subsets nominated in 4.4.

4.6 REQUIREMENTS

For each subset nominated in 4.4 the equipment shall be capable of establishing, using and terminating a session-connection by execution of the protocol according to the following criteria. The subject equipment:

"a) shall accept correct sequences of SPDUs received from peer equipment, and respond with correct SPDU sequences, for the defined states of a session connection,

b) shall respond correctly to all incorrect sequences of SPDUs received for a defined state of a session connection,

c) shall accept all correct sequence of the transport mapping protocol,

d) shall only issue correct sequences of the transport mapping protocol,

e) should be capable of issuing all correct sequences of SPDUs and transport mapping protocol".

The terms "correct sequences" and "incorrect sequences" refer to the protocol defined in section 3 and Appendix D.

NOTE 47

Only the externally visible and externally testable criteria are defined.

APPENDICES

APPENDIX A

BRIEF DESCRIPTION OF THE REFERENCE MODEL OF OPEN SYSTEMS INTERCONNECTION

A.1 SCOPE

This appendix is not an integral part of the standard. It is a copy of ISO/TC97/SC16 N 575.

This appendix provides a brief description of the Reference Model of Open Systems Interconnection.

A.2 GENERAL DESCRIPTION

A.2.1 Introduction

The Reference Model of Open Systems Interconnection provides a common basis for the co-ordination of the development of new standards for the interconnection of systems and also allows existing standards to be placed within a common framework. The model is concerned with systems comprising terminals, computers and associated devices and the means for transferring information between these systems.

A.2.2 Overall perspective

The model does not imply any particular systems implementation, technology or means of interconnection, but rather refers to the mutual recognition and support of the standardized information exchange procedures.

A.2.3 The Open Systems Interconnection environment

Open Systems Interconnetion is not only concerned with the transfer of information between systems (i.e. with communcation), but also with the capability of these systems to interwork to achieve a common (distributed) task. The objective of Open Systems Interconnection is to define a set of standards which allow interconnected systems to co-operate.

The Reference Model of Open Systems Interconnection recognizes three basic constituents (see fig. 1):

a) application processes within an OSI environment,

b) connections which permit information exchange,

c) the systems themselves.

NOTE A.1

The application processes may be manual, computer or physical processes.

A.2.4 Management Aspects

Within the Open Systems Interconnection architecture there is a need to recognize the special problems of initiating, terminating, monitoring on-going activities and assisting in their harmonious operations as well as handling abnormal conditions. These have been collectively considered as the management aspects of the Open Systems Interconnection 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.

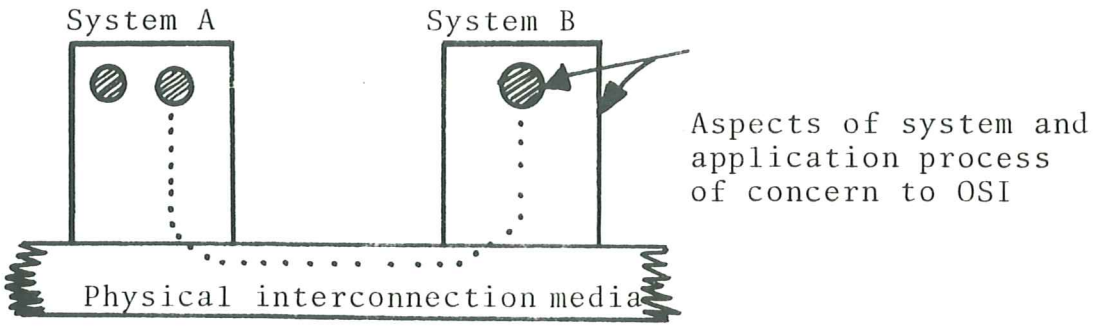


Fig. 1 - General schematic diagram illustrating the basic elements of Open Systems Interconnection

A.2.5 Concepts of a Layered Architecture

The open systems architecture is structured in Layers. Each system is composed of an ordered set of sub-systems represented for convenience by Layers in a vertical sequence. Adjacent subsystems communicate through their common interface.

A Layer consists of all subsystems with the same rank. The operation of a layer is the sum of the co-operation between entities in that Layer. It is governed by a set of protocols specific to that Layer.

The services of a Layer are provided to the next higher Layer, using the functions performed within the Layer and the services available from the next lower Layer.

An entity in a Layer may provide services to one or more entities in the next higher Layer and use the services of one or more entities in the next lower Layer.

A.3 THE LAYERED MODEL

The seven-Layer Reference Model is illustrated in fig. 2.

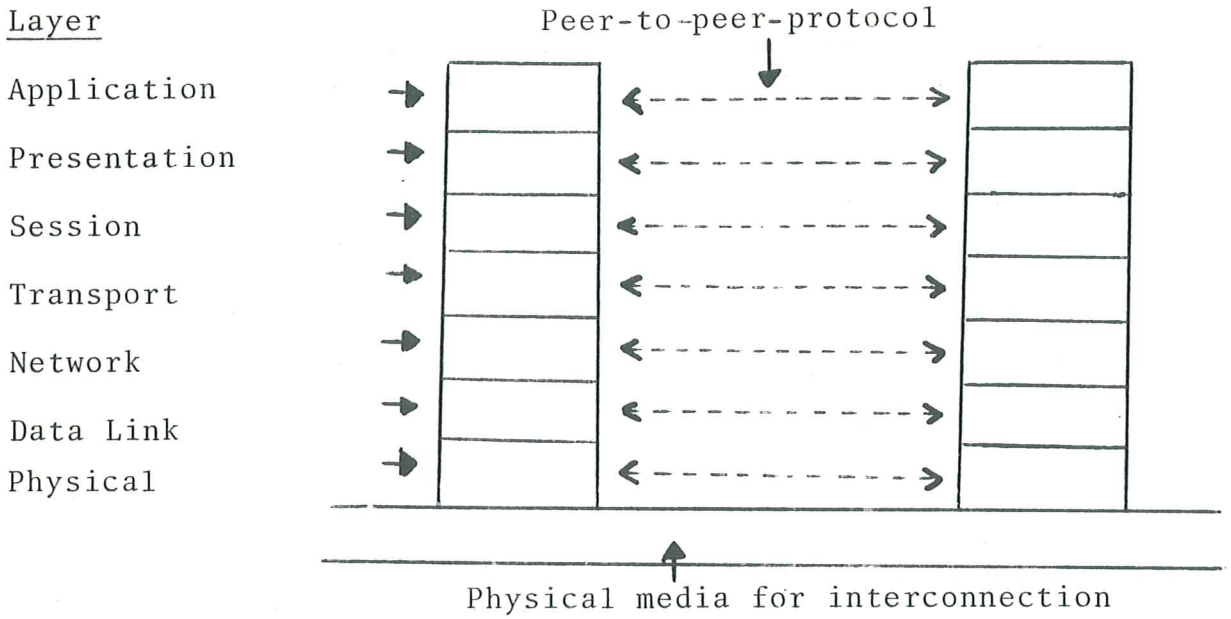


Fig. 2 - The seven-layer Reference Model and peer-to-peer protocol.

A.3.1 The Application Layer

As the highest layer in the Reference Model of Open Systems Interconnection, the Application Layer provides services to the users of the OSI environment, not to a next higher layer.

The purpose of the Application Layer is to serve as the window between communicating users of the OSI environment through which all exchange of meaningful (to the users) information occurs.

The user is represented by the application-entity to its peer.

All user specifiable parameters of each communication instance are made known to the OSI environment (and, thus, to the mechanisms implementing the OSI environment) via the Application Layer.

A.3.2 The Presentation Layer

The purpose of the Presentation Layer is to represent information to communicating application-entities in a way that preserves meaning while resolving syntax differences.

The nature of the boundary between the Application Layer and the Presentation Layer is different from the nature of other Layer boundaries in the architecture.

The following principles are adopted to define a boundary between the Application Layer and the Presentation Layer.

a) internal attributes of the virtual resource and its manipulation functions exist in the Presentation Layer;

b) external attributes of the virtual resource and its manipulation functions exist in the Application Layer;

c) the functions to use the services of the Session Layer effectively exist in the Presentation Layer;

d) the functions to use services of the Presentation Layer effectively exist in the Application Layer.

A.3.3 The Session Layer

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

A.3.4 The Transport Layer

The Transport Layer exists to provide the transport-service in association with the underlying services provided by the supporting layers.

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

The Transport Layer is required to optimize the use of the available communication resources to provide the performance required by each communicating transport user at minimum cost. This optimization will be achieved within the constraints imposed by considering the global demands of all concurrent transport users and the overall limit of resources available to the Transport Layer. Since the network service provides network connections from any transport entity to any other, all protocols defined in the Transport Layer will have end-to-end significance, where the ends are defined as the correspondent transport-entities.

The transport functions invoked in the Transport Layer to provide requested service quality will depend on the quality of the network service. The quality of the network service will depend on the way the network service is achieved.

A.3.5 The Network Layer

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

A.3.6 The Data Link Layer

The purpose of the Data Link Layer is to provide the functional and procedural means to activate, maintain and deactivate one or more data link connection among network entities.

The objective of this layer is to detect and possibly correct errors which may occur in the Physical Layer. In addition, the Data Link Layer conveys to the Network Layer the capability to request assembly of data circuits within the Physical Layer (i.e. the capability of performing control of circuit switching).

A.3.7 The Physical Layer

The Physical Layer provides mechanical, electrical, functional and procedural characteristics to activate, maintain and deactivate physical connections for bit transmission between data link entities possibly through intermediate systems, each relaying bit transmission within the Physical Layer.

APPENDIX B

TERMINOLOGY

B.1 GENERAL

This appendix is an integral part of the standard.
The terminology used in this standard consists of:

- Reference Model terminology, which is defined in ISO/DP 7498,
- Terminology for session layer concepts, services and protocol, which is defined in this appendix (see B.2),
- Notation terminology, which is defined in Appendix C.

B.2 DEFINITIONS

B.2.1 Synchronization

The effect of actions occuring at the same time.

B.2.2 Resynchronization, resynchronize

Destructive change of synchronization.

B.2.3 Transit Delay

The time taken to convey information from its source to its destination, including the time for transmission, buffering and processing.

B.2.4 Collision

Any conflict between a session service action initiated by one ss-user and an action initiated by another ss-user.

B.2.5 Contention

An actual or potential collision between two actions of the same kind.

B.2.6 Synchronization-point

A non-disruptive correlation of points in each direction of data transfer on a session connection, such that they are deemed to occur at the same time.

B.2.7 Major Synchronization Point

A synchronization point which completely isolates any subsequent communication from the previous communication.

B.2.8 Minor Synchronization Point

A synchronization point which not necessarily isolates all communications before and after it.

B.2.9 Synchronization-Point-Serial-Number

Synchronization-point identifier.

B.2.10 Dialogue-Unit

A particular kind of subdivision of activity within a session connection.

B.2.11 Token

An attribute which is dynamically assigned to one ss-user at a time or to neither ss-user.

B.2.12 Data Token

The token controlling the conditional right to initiate transfer of a normal SSDU in TWA or one-way communication.

B.2.13 Synchronize Token

The token controlling the conditional right to initiate the making of a minor synchronization point.

B.2.14 End-DU-Token

The token controlling the conditional right to initiate normal termination of a dialogue unit.

B.2.15 Terminate Token

The token controlling the conditional right to initiate negotiated termination of a session connection.

B.2.16 Delivery-Point

A point at which events subject to quarantining effects are freed from these effects.

APPENDIX C

NOTATION

C.1 INTRODUCTION AND SCOPE

This appendix is an integral part of the standard.

It is a version of notation under development in the ECMA register of common techniques for OSI standards, particularized by substitution of the word "session" into the terms defined.

C.2 DEFINITIONS

This terminology is for the notation defined in this appendix.

C.2.1 (x)-Service

A conceptual unit of the session layer service, of which (x) is its particular name.

C.2.2 (x)-Service primitive

A discrete component of an (x)-service.

C.2.3 (x)-Request primitive

A type of primitive by means of which an ss-user causes an occurence of the (x)-service.

C.2.4 (x)-Indication primitive

A type of primitive by means of which an ss-user is informed of an occurence of the (x)-service.

C.2.5 (x)-Response primitive

A type of primitive by means of which an ss-user replies to an occurence of an (x)-indication primitive.

C.2.6 (x)-Confirmation primitive

A type of primitive by means of which an ss-user is informed of an occurence of an (x)-response primitive.

C.2.7 Service Structure

The series of one or more primitives of which an (x)-service wholly consists.

C.2.8 Service Structure Type 1

A service structure with a request primitive and an indication primitive.

C.2.9 Service Structure Type 2

A service structure with a request primitive, an indication primitive, a response, and a confirmation primitive.

C.2.10 Service Structure Type 3

Service structure with two indication primitives.

C.2.11 Service Structure Type 4

Service structure with a request primitive only.

C.2.12 Event

The occurence of a primitive.

C.2.13 Initiator (ss-user)

The ss-user who issues the request primitive of the (x)-service concerned.

C.2.14 Acceptor (ss-user)

The ss-user who receives the indication primitive of the (x)-service concerned.

C.3 SERVICE MODEL

The Session Service is modelled as an abstract service to which session-service-users (ss-users) gain access at session-service-access-points (SSAPs). These SSAPs are uniquely identified by session-service-addresses. All interactions defined are between two ss-users located at separate SSAPs. A single session-connection is modelled. Figure C/1 is a pictorial representation of this model.

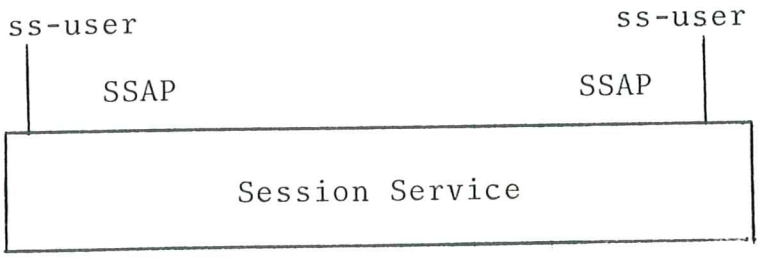


Fig. C/1: - Session service model

C.4 PRIMITIVES

The session-service is defined by means of service primitives.

Primitives are conceptual, and are not intended to be directly related to session protocol elements or to the units of interaction across a procedural interface in an implementation. The descriptive technique is independent of such variable details.

Primitives which relate only to local conventions between an ss-user and an implementation are not defined.

The subdivision of the session-service into the particular primitives chosen is arbitrary, in that the same session-service could be described by other logically equivalent primitives. There is no notion that a primitive is "elementary".

A primitive occurs at one SSAP (not both). It usually has parameters, containing values relating to its occurence.

The occurence of a primitive is a logically instantaneous and indivisible event. The primitive occurs as a logically separate instant, which cannot be interrupted by the occurence of another primitive. It occurs either completely or not at all.

There are four types of primitives in this standard:

a) request primitive (see C.2.3),
b) indication primitive (see C.2.4),
c) response primitive (see C.2.5),
d) confirmation primitive (see C.2.6).

The primitives are given names prefixed by "S" (session) to distinguish them from primitives of adjacent layers. The names of primitives are written in upper case, e.g. S-DATA.

C.5 SERVICE STRUCTURE

Each service consists of one or more primitives and affects both SSAPs, (see note). There are four different combinations of primitives. These combinations are referred to as service structures. The four service structure types used in this standard are defined in C.2.

Figures C/3, C/4, C/5 and C/6 are time-sequence diagrams giving a pictorial representation of the service structures. The progression of time is on the vertical axis downwards. The two sides represent the two ss-users. The void in the middle represents the unseen internal workings of the session service. The arrows give the direction of event propagation.

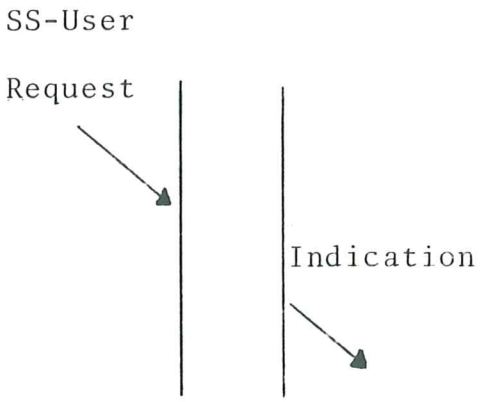


Fig. C/2: Service structure Type 1

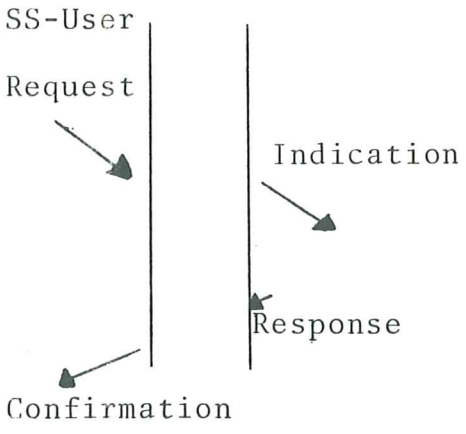


Fig. C/3: Service Structure Type 2

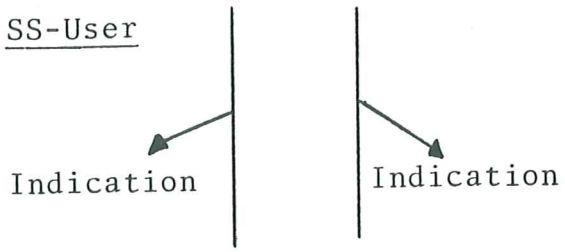


Fig. C/4: Service structure Type 3

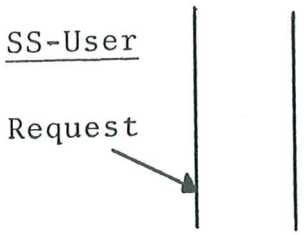


Fig. C/5: Service Structure Type 4

Unlike the events which are the occurrence of its constituent primitives, the occurence of a service structure is not logically instantaneous and indivisible. The intervals between its constituent events may be non-disruptively interspersed with other events (subject to restrictions particular to the services concerned). Service structures may also be disrupted by the occurence of certain other primitives, (see C/6).

NOTE C.1

The type 4 service structure departs from the general rule in that it has only one primitive at one SSAP. But it affects the occurence of other primitives at the other SSAP.

C.6 EFFECTS OF SERVICES

The effects of a service are referred to as being sequentially transmitted if its successive events at one SSAP result in the same sequence of corresponding events at the other SSAP, (unless disrupted, see below).

The effects of a service are referred to as being expedited if its indication or confirmation events may arrive at the other SSAP before those of a previous occurence of a service.

The effects of a service are referred to as being disruptive if it may destroy, and therefore prevent the occurence of, indication and confirmation events corresponding to previous request or confirmation events. The effects of disruptive services are expedited unless stated otherwise.

The effects of a service are referred to as being non-disruptive if they do not have the above disruptive effect. Non-disruptive effects may include effects relating to delaying of other events without destroying them.

The effects of most services are sequentially transmitted and non-disruptive.

The characteristics of individual primitives or events may also be referred to by this same terminology, (e.g. "disruptive primitive" or "expedited event").

C.7 PARAMETER NOTATION

For each service structure, the parameters are defined by a table and a numbered list of rules referencing the table. The vertical axis of each table is a list of the parameters. There is a column for each primitive. In each column there is a source code against each parameter.

D = Value supplied by ss-user during the event (D="down").
U = Value supplied by session service during the event (U="up").
B = Value supplied by ss-user and/or the session service during the event (B="both").
X = Parameter not used during the event.
- = Source not defined in this table.

The letter may be followed by a number referencing the rule which applies to the parameter.

Unless otherwise stated, the parameter value in the indication is the same as that in the request, and the parameter value in the confirmation is the same as that in the response.

Parameter values are only defined to a level which distinguishes meaning (e.g. "yes"), but generally without defining their absolute value or encoding. These details are outside the scope of the standard, being local conventions between an ss-user and an implementation, or else conventions between the ss-users which may be the subject of other standards.

The value "null" is used with the meaning that the characteristic referred to is not wanted, and includes the case where the parameter is not used.

In some cases an event has no parameters.

APPENDIX D

FORMAL DESCRIPTION

This appendix is an integral part of the standard.

D.1 INTRODUCTION

In section 3 of this standard, the SPDU protocol interactions between two SPMs are described. That description references states, events and actions which in this appendix are consolidated into a formal description of the SPDU protocol by means of state-event tables.

This is a description of the complete protocol and is independent of the subsetting within it. For some of the subsets defined, some of the information may not be applicable.

The TSM protocol is not included in this formal description.

D.2 ELEMENTS USED IN THE FORMAL DESCRIPTION

Table D/1 lists the states which are used in the formal description. For each entry there is a state code and a brief description.

Table D/2 lists the events which are used in the formal description. For each entry there is an event code and a brief description.

Table D/3 lists the conditions which are used in the formal description. For each entry there is a condition code and a brief description.

Table D/4 lists the SPM message acronyms which are used in the formal description to identify messages sent. Sometimes they are qualified by extra information in parenthesis. For each entry there is an acronym and the message name.
The way in which these various elements are used is defined in D.3.

In the Data Transfer message, concatenated events shall be processed in the sequence Data (EVE 15)
Mark (EVE 15A, 15B, 15C)
Tokens (EVE 21)
as if they were included in separate messages.

NOTE D.1

Tables D/1 and D/2 also include SPM states and SPM events which are referenced in the text description, section 3.2, but not in this formal description. This reflects differences in the level of detail between the text description and the formal description.

Table D/1 - SPM States

State Code	State Description
STA 1	Unconnected
STA 2A	waiting for ACCEPT message
STA 3	waiting for DISCONNECT message
STA 4A	waiting for MARK D CONFIRMATION message DATA TOKEN my side
STA 4B	waiting for MARK D CONFIRMATION message DATA TOKEN not my side
STA 5	waiting for RESYNCHRONIZE ACKNOWLEDGEMENT message (initiator point of view)
STA 6	waiting for RESYNCHRONIZE ACK (collision between two requests)
STA 6A	waiting for old RESYNCHRONIZE message after another PR (RS) while resolving a previous collision
STA 7	Idle DATA TOKEN my side
STA 8	waiting for ESTABLISH response event
STA 9	waiting for RELEASE response event
STA 10A	waiting for MARK D response event, DATA TOKEN my side
STA 10B	waiting for MARK D CONFIRMATION, DATA TOKEN not my side
STA 11	waiting for RESYNCHRONIZE response event
STA 12	waiting for MARK C confirmation
STA 13	Idle DATA TOKEN not my side
STA 7-13	connected
STA 15A	after PREPARE, waiting for MARK D CONFIRMATION message
STA 15B	after PREPARE, waiting for RESYNCHRONIZE message (acceptor point of view)
STA 15C	after PREPARE, waiting for RESYNCHRONIZE ACK message
STA 15D	waiting for new RESYNCHRONIZE message after another PR (RS) while resolving a previous collision

NOTE D.2

STA 1 exists only after a transport connection establishment and ceases to exist after transport connection termination.

Table D/2 - SPM Events

Event Code	Event Description
	SPM-driving events
EVE 1	ESTABLISH request event
EVE 2	ABORT request event
EVE 3	RELEASE request event
EVE 4	TRANSFER request event
EVE 4A	MARK B request event
EVE 4B	MARK D request event
EVE 4C	MARK C request event
EVE 5	CANCEL request event
EVE 6	EXPEDITED request event
EVE 7	RESYNCHRONIZE request event
EVE 7A	RESYNCHRONIZE (ABANDON) request event
EVE 7B	RESYNCHRONIZE (RESTART) request event
EVE 8	GIVE TOKENS request event
EVE 9	PLEASE TOKENS request event
EVE 10A	REJECT request event
EVE 11	CONNECT incoming message event
EVE 12	ACCEPT incoming message event
EVE 12A	REFUSE incoming message event
EVE 13	FINISH incoming message event
EVE 14	DISCONNECT incoming message event
EVE 15	DATA TRANSFER incoming message event
EVE 15A	MARK B incoming message event
EVE 15B	MARK D incoming message event
EVE 15C	MARK C incoming message event
EVE 16	MARK CONFIRMATION incoming message event
EVE 16A	MARK B CONFIRMATION incoming message event
EVE 16B	MARK D CONFIRMATION incoming message event
EVE 16C	MARK C CONFIRMATION incoming message event
EVE 17	CANCEL incoming message event
EVE 18	EXPEDITED incoming message event
EVE 19	RESYNCHRONIZE incoming message event
EVE 20	RESYNCHRONIZE ACK incoming message event

Table D/2 - continued

EVE 21	GIVE TOKENS incoming message event
EVE 23	PLEASE TOKENS incoming message event
EVE 31	ABORT incoming message event
EVE 32	PREPARE incoming message event
EVE 32A	PREPARE (MARK D) incoming message event
EVE 32B	PREPARE (RESYNC) incoming message event
EVE 32C	PREPARE (RESYNC ACK) incoming message event
EVE 33	NOT FINISHED incoming message event
EVE 25	ESTABLISH reponse event
EVE 26A	RELEASE AFFIRMATIVE response event
EVE 26B	RELEASE NEGATIVE response event
EVE 27	MARK CONFIRMATION response event
EVE 27A	MARK B/C CONFIRMATION response event
EVE 27B	MARK D CONFIRMATION response event
EVE 28	RESYNCHRONIZE ACK response event
	SPM generated events
EVE 101	ESTABLISH indication event
EVE 102	ABORT indication event
EVE 103	RELEASE indication event
EVE 104	TRANSFER indication event
EVE 104A	MARK B/C indication event
EVE 104B	MARK D indication event
EVE 106	EXPEDITED indication event
EVE 107	RESYNCHRONIZE indication event
EVE 108	GIVE TOKENS indication event
EVE 109	PLEASE TOKENS indication event
EVE 110A	REJECT indication event
EVE 112	ESTABLISH confirmation event
EVE 113	MARK confirmation event
EVE 113A	MARK B/C confirmation event
EVE 113B	MARK D confirmation event
EVE 114	RESYNCHRONIZE confirmation event
EVE 116A	RELEASE AFFIRMATIVE confirmation event
EVE 116B	RELEASE NEGATIVE confirmation event

Table D/3 Conditions

Conditions	Meaning
DT	DATA TOKEN my side
^DT	DATA TOKEN not my side
SY	SYNCHRONIZE TOKEN my side
^SY	SYNCHRONIZE TOKEN not my side
DU	END-DU-TOKEN my side
^DU	END-DU-TOKEN not my side
TR	TERMINATE-TOKEN my side
^TR	TERMINATE-TOKEN not my side
DEF (xx)	TOKEN XX is defined
^DEF (xx)	TOKEN XX is not defined

Table D/4 List of SPM messages

CONNECT	CN
ACCEPT	AC
REFUSE	RF
FINISH	FN
NOT FINISHED	NF
DISCONNECT	DN
ABORT	AB
DATA TRANSFER	DT
MARK CONFIRMATION	MC
EXPEDITED	EX
CANCEL	CL
RESYNCHRONIZE	RS
RESYNCHRONIZE ACKNOWLEDGEMENT	RA
PLEASE TOKENS	PT
GIVE TOKENS	GT
PREPARE	PR

D.3 FORMAL DESCRIPTION CONVENTIONS

The formal description is in tables D/6 to D/17.

The horizontal dimension of each table is the set of all the states. If for any given state there is no valid event, then the state does not appear in the table, i.e. no column (see note D.3).
The vertical dimension of each table is the set of all relevant SPM request events, incoming message event and SPM response events. For each event there is an entry, i.e. a row.

Each valid intersection contains:

- one or more conditions (where relevant)
- one or more actions (where relevant);
- the new state (always).

The conditions are those defined in table D/3.

The action generally consists of sending a message or issuing a local primitive event (indication event or confirmation event). Sometimes the event causing an action is locally enqueued, or causes other events to be dequeued or cancelled. This is indicated by a store, recall or cancel code:

- "ST" means store (enqueue);
- "RC" means recall (dequeue);
- "CL" means cancel (purge queue).

The new state is the state which is entered after the specified action is completed. All dequeued events are considered by the SPM with priority over any other incoming events. An invalid intersection is shown by a stroked entry (see note).

The applicability of some tables depends on whether certain tokens are defined or not defined. Any such constraint is explicitly stated.

All invalid intersection between states and incoming message events are treated as protocol errors. The actions are: issue EVE 102 ABORT indication event if the transport connection is useable, then send AB message with reason code = protocol error. The new state is STA 1 unconnected.

NOTE D.3

A state which does not appear in a given table may appear in another table which has different applicability. Likewise for an intersection, which is invalid in a given table may be valid in another table which has different applicability.

D.4 FORMAL DESCRIPTION TABLES

The session protocol is described by using several tables: this introduces redundancy but allows individual description of each sub-protocol. For editing reasons, in some

cases the same table cannot contain all the states: then consecutive tables are used.

In order to know how a SPM should behave when it is in a given state S_i and when a given event E_j occurs, find the table(s) with the appropriate applicability, by using the index in table D/5. If the appropriate table contains the state S_i and the event E_j, the action and the new state etc. are defined in the intersection (S_i, E_j). Otherwise the combination (S_i, E_j) is invalid.

Table D/5 - Index of Formal Description Tables

Table	Sub-protocol	Codes for applicability condition
D/6	Connection	None
D/7	Data Tranfer	DEF (DT)
D/8	Data Transfer	DEF (DT), DEF (TR)
D/9	Data Transfer	DEF (DT), ^DEF (TR)
D/10	Data Transfer	^DEF (DT)
D/10-1	Data Transfer	^DEF (DT)
D/11	Mark B/C	None
D/11-1	Mark B/C	None
D/12	Mark D	None
D/13	Resynchronize	None
D/13-1	Resynchronize	None
D/14	Disconnection	DEF (TR)
D/15	Disconnection	^DEF (TR)
D/16	Abort	None
D/16-1	Abort	None
D/17	Token Transfer	None
D/17-1	Token Transfer	None

NOTE D.4

The notation in the last column of the table D/5 is defined in table D/3.

Table D/6 Connection

STATES / EVENTS	STA 1 UNCONNECT	STA 2A wait for ACCEPT message	STA 8 wait for ESTABLISH response
EVE 1 ESTABLISH request	Send CN STA 2A		
EVE 25 ESTABLISH response			Send AC STA 7-13
EVE 10A REJECT request			Send RF STA 1
EVE 11 CONNECT message	EVE 101 ESTABLISH indic. STA 8		
EVE 12 ACCEPT message		EVE 112 ESTABLISH confir. STA 7-13 RC	
EVE 12A REFUSE message		EVE 110A REJECT indication STA 1	

Table D/7 Data Transfer DEF (DT)

STATES / EVENTS	STA 2A wait for ACCEPT message	STA 4 wait for MARK D CONF mess.	STA 5 wait for RESYN ACK mess.	STA 7-13 CONNECTED	STA 12 wait for MARK C confir.	STA 15A wait after PR (MARK D CONF) mess.	STA 15B wait after PR (RESYN) message
EVE 4 TRANSFER request				DT condit. send DT STA 7-13			
EVE 6 EXPEDITED request				send EX STA 7-13	Send EX STA 12		
EVE 5 CANCEL request				DT cond. send CL STA 7-13			
EVE 15 DATA TRANSFER message			STA 5	^DT cond EVE 104 Transfer indication STA 7-13			STA 15B
EVE 18 EXPEDITED message	STA 2A ST	EVE 106 EXPEDITED indication STA 4	STA 5	EVE 106 EXPEDITED indication STA 7-13	EVE 106 EXPEDITED indication STA 12	STA 15A ST	STA 15B
EVE 17 CANCEL message			STA 5	^DT cond. cancella-tion STA 7-13			STA 15B

Table D/8 Data Transfer DEF (DT),DEF (TR)

STATES / EVENTS	STA 3 wait for DISCONNECT message	STA 6 wait for RESYN ACK message (collision)	STA 15C wait after PR (RESYN ACK) mess.
EVE 4 TRANSFER request			
EVE 6 EXPEDITED request			
EVE 5 CANCEL request			
EVE 15 DATA TRANSFER message		STA 6	STA 15C
EVE 18 EXPEDITED message	EVE 106 EXPEDITED indication STA 3	STA 6 ST	STA 15C ST
EVE 17 CANCEL message		STA 6	STA 15C

Table D/9 Data Transfer DEF (DT),^DEF (TR)

STATES / EVENTS	STA 3 wait for DISCONNECT message	STA 6 wait for RESYN ACK message (collision)	STA 15C wait after PR (R1) message
EVE 4 TRANSFER request			
EVE 6 EXPEDITED request			
EVE 5 CANCEL request			
EVE 15 DATA TRANSFER message	^DT cond . EVE 104 TRANSFER indication STA 3	STA 6	STA 15C
EVE 18 EXPEDITED message	EVE 106 EXPEDITED indication STA 3	STA 6 ST	STA 15C ST
EVE 17 CANCEL message	^DT cond cancella-tion STA 3	STA 6	STA 15C

Table D/10 Data Transfer ^DEF (DT)

STATES / EVENTS	STA 2A wait for ACCEPT message	STA 3 wait for DISCONNECT message	STA 4 wait for MARK D CONF mess.	STA 5 wait for RESYN ACK message	STA 6 wait for RESYN ACK message (collision)
EVE 4 TRANSFER request					
EVE 6 EXPEDITED request					
EVE 5 CANCEL request					
EVE 15 DATA TRANSFER message		EVE 104 TRANSFER indication STA 3	EVE 104 TRANSFER indication STA 4	STA 5	STA 6
EVE 18 EXPEDITED message	STA 2A ST	EVE 106 EXPEDITED indication STA 3	EVE 106 EXPEDITED indication STA 4	STA 5	STA 6ST
EVE 17 CANCEL message		cancella-tion STA 3	cancella-tion STA 4	STA 5	STA 6

Table D/10-1 Data Transfer ^DEF (DT)

STATES / EVENTS	STA 7-13 CONNECTED	STA 10 wait for MARK D response	STA 12 wait for MARK C CONF mess.	STA 15A wait after PR (MARK D CONF) mess.	STA 15B wait after PR (RESYN message	STA 15C wait after PR (RESYN ACK) mess.
EVE 4 TRANSFER request	send DT STA 7-13	send DT STA 10			STA 15B	
EVE 6 EXPEDITED request	send EX STA 7-13	send EX STA 10	send EX STA 12		STA 15B	
EVE 5 CANCEL request	send CL STA 7-13	send CL STA 10			STA 15B	
EVE 15 DATA TRANSFER message	EVE 104 TRANSFER indication STA 7-13		EVE 104 TRANSFER indication STA 12	EVE 104 TRANSFER indication STA 15A	STA 15B	STA 15C
EVE 18 EXPEDITED message	EVE 106 EXPEDITED indication STA 7-13		EVE 106 EXPEDITED indication STA 12	STA 15A ST	STA 15B	STA 15C ST
EVE 17 CANCEL message	cancella-tion STA 7-13		cancella-tion STA 12	cancella-tion STA 15A	STA 15B	STA 15C

Table D/11 Mark B/C

STATES / EVENTS	STA 3 wait for DISCONNECT message	STA 4 wait for MARK D CONF mess.	STA 5 wait for RESYN ACK message	STA 6 wait for RESYN ACK message	STA 7-13 CONNECTED	STA 9 wait for RELEASE response
EVE 4A MARK B request					DT SY cond. send DT (MARK B) STA 7-13	
EVE 4C MARK C request					DT SY cond send DT (MARK C) CONFIRM STA 12	
EVE 27A MARK B/C CONFIRM response					send MC STA 7-13	send MC STA 9
EVE 15A MARK B/C message	^DEF (DT) ^SY cond EVE 104A MARK B/C indication STA 3		STA 5	STA 6	^DT ^SY cond. EVE 104A MARK B/C indication STA 7-13	
EVE 16A MARK B CONFIRM message	EVE 113A MARK B/C CONFIRM STA 3	EVE 113A MARK B/C CONFIRM STA 4	STA 5	STA 6	EVE 113A MARK B/C CONFIRM STA 7-13	
EVE 16C MARK C CONFIRM message			STA 5	STA 6		

NOTE D.5

The Synchronize Token is defined.
Also, if the Data Token is not defined, ignore the DT and ^DT conditions in this table.

Table D/11-1 Mark B/C

STATES / EVENTS	STA 10 wait for MARK D response	STA 12 wait for MARK C CONF mess.	STA 15A wait after PR (MARK D CONF) mess.	STA 15B wait after PR (RESYN) message	STA 15C wait after PR (RESYN ACK) mess
EVE 4A MARK B request				STA 15B	
EVE 4C MARK C request				STA 15B	
EVE 27A MARK B/C CONFIRM response	send MC STA 10			STA 15B	
EVE 15A MARK B/C message				STA 15B	STA 15C
EVE 16A MARK B CONFIRM message		EVE 113A MARK B/C CONFIRM STA 12	EVE 113A MARK B/C CONFIRM STA 15A	STA 15B	STA 15C
EVE 16C MARK C CONFIRM message		EVE 113A MARK B/C CONFIRM STA 7-13		STA 15B	STA 15C

NOTE D.5

The Synchronize Token is defined.
Also, if the Data Token is not defined, ignore the DT and `DT conditions in this table.

Table D/12 Mark D

STATES / EVENTS	STA 3 wait for DISCONNECT message	STA 4 wait for MARK D CONF mess.	STA 5 wait for RESYN ACK message	STA 6 wait for RESYN ACK (coll.)	STA 7-13 CONNECTED	STA 10 wait for MARK D response	STA 15A wait after PR(MARK D CONF) mess.	STA 15B wait after PR (RESYN) message	STA 15C wait for RESYN ACK message
EVE 4B MARK B request					DT, SY and DU cond send DT (MARK D) STA 4			STA 15B	
EVE 27B MARK D CONFIRM response						send PR (MARK D CONF)mess. send MC STA 7-13		STA 15B	
EVE 15B MARK D message	^SY ^DEF (DT) and ^DU cond EVE 104B MARK D ind. STA 3		STA 5	STA 6	^DT, ^SY and ^DU cond EVE 104B MARK D ind. STA 10			STA 15B	STA 15C
EVE 32A PR MARK D CONFIRM message		STA 15A	STA 5						
EVE 16B MARK D CONFIRM message			STA 5	STA 6			EVE 113B MARK D CONFIRM STA 7-13 RC		STA 15C

NOTE D.6

The End-DU Token is defined.
Also, if the Data Token is not defined ignore the DT and ^DT conditions in this table.
Also, if the Synchronize Token is not defined, ignore the SY and ^SY conditions in this table.

Table D/13 Resynchronize

STATES / EVENTS	STA 2A wait for ACCEPT message	STA 3 wait for DISCONNECT message	STA 4 wait for MARK D conf. mess.	STA 5 wait for RESYN ACK message	STA 6 wait for RESYN ACK (collision)	STA 7-13 CONNECTED	STA 9 wait for RELEASE reponse
EVE 7B RESYN (restart) request						send PR (RESYN) send RS STA 5	send PR (RESYN) send RS STA 5
EVE 7A RESYN (abandon) request			send PR (RESYN) send RS STA 5			send PR (RESYN) send RS STA 5	send PR (RESYN) send RS STA 5
EVE 28 RESYN ACK resp							
EVE 32B PREPARE RESYN message	STA 2A ST	STA 15B	STA 15B	STA 6	STA 6A	STA 15B	
EVE 19 RESYN message					RS winner STA 5 RC RS winner EVE 107 RESYN ind STA 11		
EVE 32C PREPARE RESYN ACK message				STA 15C	STA 6 ST		
EVE 20 RESYN ACK message							

NOTE D.7

Condition "RS-Winner" is verified if and only if the previous RESYN to which the SPM was prepared (waiting either for an acknowledgement or a response) prevails over the new event, according to the contention resolution rules (see 3.2.5.2).

Table D/13-1 Resynchronize

STATES / EVENTS	STA 10 wait for MARK D response	STA 11 wait for RESYN response	STA 12 wait for MARK C CONF mess.	STA 15A wait after PR (MARK D CONF)mess.	STA 15B wait after PR (RESYN) message	STA 15C wait after PR (RESYN) ACK) mess.	STA 6A wait for old RESYN after another PR (RS)	STA 15D wait for new RESYN after another PR (RS)
EVE 7B RESYN (restart request)	send PR (RESYN) send RS STA 5	^RS winner send PR (RESYN) send RS STA 5	send PR (RESYN) send RS STA 5		send PR (RESYN) send RS STA 6			
EVE 7A RESYN (abandon) request	send PR (RESYN) send RS STA 5	^RS winner send PR, RS RESYN STA 5	send PR (RESYN) send RS STA 5	send PR (RESYN) send RS STA 5	send PR (RESYN) send RS STA 6			
EVE 28 RESYN ACK response		send PR (RESYN ACK) send RA STA 7-13						
EVE 32B PREPARE RESYN message	STA 15B		STA 15B	STA 15B		STA 15B CL		
EVE 19 RESYN message					EVE 107 RESYN indic. STA 11		RS winner STA 15D	^RS winner EVE 107 RESYN indic. STA 11
EVE 32C PREPARE RESYN ACK message								
EVE 20 RESYN ACK message					STA 15B	EVE 114 RESYN confirm STA 7-13 RC		

NOTE D.7

Condition "RS-Winner" is verified if and only if the previous RESYN to which the SPM was prepared (waiting either for an acknowledgement or a response) prevails over the new event, according to the contention resolution rules (see 3.2.5.2).

Table D/14 Disconnection DEF (TR)

STATES / EVENTS	STA 3 wait for DISCONNECT message	STA 5 wait for RESYN ACK message	STA 6 wait for RESYN ACK (collision)	STA 7-13 CONNECTED	STA 9 wait for RELEASE response	STA 12 wait for MARK C confirm	STA 15B wait after PR (RESYN) message	STA 15C wait after PR (RESYN ACK) mess.
EVE 3 RELEASE request				DT and TR cond. send FN STA 3			STA 15B	
EVE 26A RELEASE AFFIRMAT response					send DN STA 1			
EVE 26B RELEASE NEGATIVE response					send NF STA 7-13			
EVE 13 FINISH message		STA 5	STA 6	^DT and ^TR cond EVE 103 RELEASE indic. STA 9		^ DT ^TR cond EVE 103 RELEASE indic. STA 9		STA 15C
EVE 14 DISCONNECT message	EVE 116A RELEASE AFFIRMAT confirm STA 1							
EVE 33 NOT FINISHED message	EVE 116B RELEASE NEGATIVE confirm STA 7-13						EVE 116B RELEASE NEGATIVE confirm STA 15B	

NOTE D.8

If the Data Token is not defined, ignore the DT and ^DT conditions in this table.

Table D/15 Disconnection ^DEF (TR)

STATES / EVENTS	STA 1 UNCONNECTED	STA 3 wait for DISCONNECT message	STA 4 wait for MARK D CONF mess.	STA 5 wait for RESYN ACK message	STA 6 wait for RESYN ACK (collision)	STA 7-13 CONNECTED	STA 9 wait for RELEASE response	STA 12 wait for MARK C CONF mess.	STA 15C wait after PR(RESYN ACK)mess.
EVE 3 RELEASE request						DT cond send FN STA 3			
EVE 26A RELEASE AFFIRMAT response							send DN STA 1		
EVE 26B RELEASE NEGATIVE response									
EVE 13 FINISH message		EVE 103 RELEASE indic. STA 9	EVE 103 RELEASE indic. STA 9	STA 5	STA 6	^DT cond. EVE 103 RELEASE indic. STA 9		^DEF(DT) cond. EVE 103 RELEASE indic. STA 9	STA 15C
EVE 14 DISCONN message	STA 1	EVE 116A RELEASE AFFIRMAT confirm STA 1					EVE 116A RELEASE AFFIRM confirm STA 9		
EVE 33 NOT FINISHED message									

Table D/16 Abort

STATES / EVENTS	STA 1 UNCONN	STA 2A wait for ACCEPT message	STA 3 wait for DISCONN message	STA 4 wait for MARK D CONF mess.	STA 5 wait for RESYN ACK message	STA 6 wait for RESYN ACK (collision)	STA 7-13 CONNECTED	STA 8 wait for ESTABL resp.	STA 9 wait for RELEASE response
EVE 2 ABORT request		Send AB STA 1	Send AB STA 1	send AB STA 1	send AB STA 1	send AB STA 1	send AB STA 1	send AB STA 1	send AB STA 1
EVE 31 ABORT message	STA 1	EVE 102 ABORT indic. STA 1	EVE 102 ABORT indic. STA 1	EVE 102 ABORT indic. STA 1	EVE 102 ABORT indic. STA 1	EVE 102 ABORT indic. STA 1	EVE 102 ABORT indic. STA 1	EVE 102 ABORT indic. STA 1	EVE 102 ABORT indic. STA 1

Table D/16-1 Abort

STATES / EVENTS	STA 10 wait for MARK D response	STA 11 wait for RESYN response	STA 12 wait for MARK C CONF mess.	STA 15A wait after PR (MARK D CONF) mess.	STA 15B wait after PR (RESYN) message	STA 15C wait after PR (RESYN ACK) mess	STA 6A wait for old RESYN after another PR (RS)	STA 15D wait for new RESYN after another PR (RS)
EVE 2 ABORT request	send AB STA 1	send AB STA 1	send AB STA 1	send AB STA 1	send AB STA 1	send AB STA 1	send AB STA 1	send AB STA 1
EVE 31 ABORT message	EVE 102 ABORT indic. STA 1	EVE 102 ABORT indic. STA 1	EVE 102 ABORT indic. STA 1	EVE 102 ABORT indic. STA 1	EVE 102 ABORT indic. STA 1	EVE 102 ABORT indic. STA 1	EVE 102 ABORT indic. STA 1	EVE 102 ABORT indic. STA 1

Table D/17 Token Transfer

STATES / EVENTS	STA 3 wait for DISCONN message	STA 4 wait for MARK D CONF mess.	STA 5 wait for RESYN ACK message	STA 6 wait for RESYN ACK (collision)
EVE 9 PLEASE TOKENS request				
EVE 8 GIVE TOKENS request				
EVE 23 PLEASE TOKEN message	EVE 109 PLEASE T indic. STA 3	EVE 109 PLEASE T indic. STA 4	STA 5	STA 6
EVE 21 GIVE TOKENS message	EVE 108 GIVE T indic. STA 3	EVE 108 GIVE T indic. STA 4	STA 5	STA 6

NOTE D.9
If the Data Token is either my side or undefined, then the action is send DT (GIVE TOKENS).

Table D/17-1 Token Transfer

STATES / EVENTS	STA 7-13 CONNECTED	STA 10 wait for MARK D response	STA 12 wait for MARK C CONF mess.	STA 15A wait after PR (MARK D CONF) mess.	STA 15B wait after PR (RESYN) message	STA 15C wait after PR (RESYN ACK) mess.
EVE 9 PLEASE TOKENS request	send PT STA 7-13	send PT STA 10			STA 15B	
EVE 8 GIVE TOKENS request	send GT (see note) STA 7-13	send GT STA 10			STA 15B	
EVE 23 PLEASE TOKEN message	EVE 109 PLEASE T indic STA 7-13		EVE 109 PLEASE T indic. STA 12	EVE 109 PLEASE T indic. STA 15A	STA 15B	STA 15C
EVE 21 GIVE TOKENS message	EVE 108 GIVE T indic STA 7-13		EVE 108 GIVE T indic. STA 12	EVE 108 GIVE T indic. STA 15A	STA 15B	STA 15C

NOTE D.9
If the Data Token is either my side or undefined, then the action is send DT (GIVE TOKENS).

APPENDIX E

EXTENSIONS

E.1 GENERAL

This appendix is not an integral part of the standard.

It is provided to aid understanding and use of the standard by explaining some future extensions and related matters which have been considered by ECMA during its development.

This is a basic session protocol, designed to allow such extension to be added in a modular way which preserves compatibility.

E.2 LIAISON WITH OTHER STANDARDIZATION BODIES

During the development of this standard by ECMA, there has been close liaison with several other standards bodies, especially the ISO Open System Interconnection Sub-committee (ISO/TC97/SC16).

ECMA is an active contributor to the development of ISO session layer standards by SC16/WG-6. The technical content of this ECMA standard has a high degree of commonality with the current (1981) ISO work, which is till at a formative stage prior to agreement of ISO standards. The intention is that future developments will continue to benefit from this close liaison.

E.3 ADDITIONAL SUBSETS

The standard defines general session layer concepts, services and protocol, and then uses these in the definition of subsets, which are the definitive items for purposes of conformance and variety control.

The subsets defined in this basic standard are designed for certain specific requirements (see 2.3). The concepts, service and protocol defined in the standard are limited to what is needed for these subsets.

When other requirements have been agreed in sufficient detail and with sufficient generality to be standardized, it will be possible to define additional standard subsets to meet these requirements, and to extend the concepts, services and protocol accordingly.

For example, it is anticipated that future development of the more sophisticated classes of virtual terminal protocol will require a richer set of session services and protocol.

E.4 FLEXIBILITY FOR FUTURE EXTENSION

The flexibility for future modular extension while preserving compatibility is obtained in three main ways.

Firstly, the concepts, services and protocol included in this initial session layer standard have been selected to support this purpose. They are sufficiently open-ended to allow future development.

Secondly, the definition of four different aspects of the protocol has been decoupled:

- the protocol for exchange of session-protocol-data-units (see subsection 3.2)
- the protocol for transport-service-mapping (see subsection 3.3.)
- the encoding of session-protocol-data-units (see subsection 3.4)
- the protocol subsets (see subsection 3.5)

To a certain degree, each of these can be changed independently, without affecting the others.

Thirdly the encoding is designed to be highly flexible:

- all variables within an element of protocol are individually fully encoded wherever practicable (instead of collective encoding of groups of variables, restricting the possible combinations),
- self-defining formats are used to enable future insertion of new variables and extension of existing ones,
- where future extensions are already anticipated, reserved fields and reserved values have been provided for them.

These encoding characteristics allow future protocol changes with least possible impact on the then existing implementations.

E.5 PERFORMANCE

Performance efficiency is an important consideration in the design of the protocol, the primary concern has been to limit the quantity of interactions needed to achieve a particular purpose and the principal means of achieving better efficiency are:

- mapping the most frequently occuring elements (i.e. data transfer, data delimiting, token transfer and marks) into one SPDU and optimizing the encoding accordingly (i.e. the DT).
- minimizing the number of "handshakes" needed in certain frequently occuring actions (e.g. connection-establishment, connection-termination, synchronization-points).
- provision for blocking of multiple SSDUs into an SPDU.

E.6 COMPLETENESS OF THE SERVICE DESCRIPTION

The definition of the abstract session-service (see section 2) provides a detailed specification of the service primitives and their parameters. It also defines the sequences of primitives within each service structure, and the general interactions between service structures (i.e. whether effects are destructive or non-destructive, normal or expedited). However, it does not define the complete detail of how all primitives interact at an SSAP (i.e. what primitives can be issued in which conditions, and what happens when certain events collide).

The definitive specification of interactions is only in the formal description of the protocol (see appendix D).

Some preliminary work has been done on comprehensive formal description of the abstract services, additional to and separate from the formal description of the protocol. This involves the development of a new notation for formal description of services (the issues are general to all services not particular to the session service). This work is not yet ready for standardization; it is a candidate for inclusion in future versions of the standard.

E.7 ADDITIONAL FACILITIES

The standard does not include session layer fast-connect/disconnect and connectionless-data-transmission facilities. The inclusion of these is for further study. Some preliminary design has been studied.

No other requirements for major additional facilities have been identified (except re-negotiation, see E.8). All the other extensions are refinements of facilities already included in the standard.

E.8 NEGOTIATION ENHANCEMENTS

In this standard, session-connection parameters are negotiated in a very simple way. At session-connection establishment, the characteristics are mostly determined by the initiator (the acceptor choice is essentially to accept or reject the connection). There is no means to change the session-connection-parameters during the lifetime of a session-connection.

ECMA has developed more sophisticated negotiation procedures for other OSI protocols. The current design is a proper subset of these procedures. Preliminary designs for such enhancements (including renegotiation of session-connection parameters) have been studied and certain encodings have been reserved for this use.

E.9 SYNCHRONIZATION ENHANCEMENTS

Preliminary designs have been studied for extension of the synchronization concepts, services and protocol defined in this standard. Encodings have been reserved for this use.

The main topics are: provision of additional subtypes of synchronization-points and marks, extra handshakes, various kinds of quiescence, "brackets", and concurrent marking of synchronization points independently in each direction of flow.

E.10 TOKENS ENHANCEMENTS

It is foreseen that some requirements for additional types of token and additional kinds of token manipulation could arise.

Additonal tokens are probably needed to support enhanced session services. There may be a need for "free-tokens" manipulated by the session-service, but with semantics defined by the ss-users transparently to the session service.

There may be a need to provide an "unassigned" state and an "assigned both ends" state for tokens and the means to move them between these and other states.

Some preliminary designs have been studied and some encodings have been reserved for these extensions.

E.11 TRANSPORT MAPPING ENHANCEMENTS

The standard defines a very simple transport mapping. Candidate enhancements are:

a) use of "fast-connect/disconnect" and "connectionless" transport services;

b) a variant which optionally avoids use of the transport expedited data transfer service, enabling use of the lowest transport protocol classes (this probably involves a restricted session service);

c) protocol for the use of multiple consecutive transport-connections by the same-session connection;

d) associated with the above is protocol to give resilience to transport failures;

e) protocol to enable the same transport-connection to be reused by multiple consecutive session-connections;

f) protocol to allow session-connection establishment in either direction on a transport connection;

g) protocol to use the data parameters of the transport connection-establishment phase (i.e synchronous establishment of session-connection and transport-connection; this depends mainly on clarification of size variables).

Some preliminary designs have been studied (e.g. use of an ABORT ACK message), and some encodings have been reserved (e.g. the transport disconnection code in FINISH, DISCONNECT and ABORT messages).

E.12 CONFORMANCE TESTING

This standard does not specify detailed testing methods to verify conformance. The technical work to define such testing methods is as yet only at a formative stage. It is intended to specify conformance testing and associated diagnostic aids in a future version of the standard.