ECMA

EUROPEAN COMPUTER MANUFACTURERS ASSOCIATION

STANDARD ECMA-93

DISTRIBUTED APPLICATION FOR MESSAGE INTERCHANGE

(MIDA)

September 1984

ECMA

EUROPEAN COMPUTER MANUFACTURERS ASSOCIATION

STANDARD ECMA-93

DISTRIBUTED APPLICATION FOR MESSAGE INTERCHANGE

(MIDA)

September 1984

BRIEF HISTORY

This ECMA Standard is one of a set of standards for Open Systems Interconnection. Open Systems Interconnection standards are intended to facilitate homogeneous interconnection between heterogeneous information processing systems. The Standard is within the framework for the coordination of standards for Open Systems Interconnection which is defined in ISO 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 and national standard 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 systems to be interconnected.

In the interest of a rapid and effective standardisation, 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.

It has been an objective in the preparation of this Standard that there should exist maximum commonality of service definitions and protocol encoding between this Standard and the corresponding CCITT Recommendations of the X.400 series, so that interworking problems are minimized.

Adopted as Standard ECMA-93 by the General Assembly of June 28 - 29, 1984.

TABLE OF CONTENTS

TABLE OF CONTENTS (cont'd)

1. GENERAL

1.1 Scope

This ECMA Standard Distributed Application for Message Interchange (MIDA):

- defines a MIDA model based on store and forward capabilities for asynchronous communication between an undetermined number of users. (See Section 2);
- defines operations on the MIDA model as abstract interactions between the MIDA service users and the MIDA service provider. (See Section 3) ;
- defines the protocols to support the above services and their mapping onto the underlying service. (See Sections 4);
- specifies the requirements for conformance with these protocols. (See Section 5).

This Standard defines what is needed for a basic message transfer and access to MIDA. It provides the consistent technical basis for further MIDA protocol standards with extended scope.

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

This Standard is for a distributed application located in the Application Layer of Open Systems Interconnection (see ISO 7498) and includes a presentation definition for this application.

1.2 Field of Application

The CCITT Recommendations of the X.400 series describe a public message handling service, the protocols to be applied between public administration management domains, and the protocols by which a private mangement domain may connect to an administration domain.

This MIDA Standard describes extensions to the CCITT Recommendations of the X.400 series with the following objectives:

- to permit the creation of multivendor management domains (public or private) by defining inter-equipment extensions to the CCITT inter-domain protocol;
- to allow free interchange of messages between MIDA Users and Users connected to an administration domain;
- to provide a topological addressing mechanism for use within management domains;
- to define mechanisms for handling undeliverable messages within a management domain;
- to enable additional functionality within a management domain, including transfer of structured documents (ODIF);
- to allow a choice of lower level mappings within a management domain;
- to enable the above functions to be supported between two MIDA domains by direct connection;

- to enable the above functions to be supported as far as possible, between two MIDA domains connected via one or more administration domains;
- to enable the free and open interconnection of MIDA systems, without the need for bilateral agreements.

1.3 References

ECMA-6	7 Bit Input/Output Coded Character Set
ECMA-75	Session Protocol
ECMA-84	Data Presentation Protocol
ECMA-xx	Office Document Interchange Format (ODIF)
ECMA-zz	MIDA General Access Service
ISO 3166	Codes for the representation of names of countries
ISO 6937	Coded Character Set for Text Communication
ISO 7498	Data Processing Open Systems Interconnection Basic Reference Model
ISO DIS 8326	Basic Connection Oriented Session Service Definition
CCITT Rec. V3	International Alphabet no.5
CCITT Rec. X.121	International Numbering Plan for PDN
CCITT Rec. X.400	Message Handling Systems: System Model - Service Elements
CCITT Rec. X.409	Message Handling Systems: Presentation Transfer Syntax and Notation
CCITT Rec. X.410	Message Handling Systems: Remote Operations and Reliable Transfer Server
CCITT Rec. X.411	Message Handling Systems: Message Transfer Layer
CCITT Rec. X.420	Message Handling Systems: Interpersonal Messaging User Agent Layer

1.4 General Overview

This overview states the general objectives and the principles of the Distributed Application for Message Interchange .

1.4.1 General Objectives

This MIDA Standard allows, in conjunction with other ECMA Standards, message handling systems to be interconnected.

This Application Layer Standard makes a full use of the OSI Reference Model and of its standards:

- the message interchange application may be easily distributed among several open systems which implement MIDA protocols;

- the need for new protocols has been minimized by re-using existing protocols wherever possible;

- MIDA systems may be interconnected without any assumption on the data networks which could be available : leased lines or public data networks, for example, may support this application.

1.4.2 MIDA Principles

MIDA systems offer asynchronous communication between an undetermined number of users and provide for store and forward capabilities.

Each MIDA system needs the existence of a directory service. This directory contains complete information about all users within the system. It also contains names of the remote MIDA entities and the addresses at which they are reachable. However directory services are outside the scope of these MIDA standards.

MIDA systems do not make any assumption on the user information to be transferred. This version of the MIDA standards allows a number of modes of user information transfer:

- CCITT - recommended types;

- ODIF;

- Transparent (octet string);

- ISO 6937 Text.

The creation, rendition, significance and management of the user information carried by MIDA systems is outside the scope of this MIDA standard.

2. MESSAGE INTERCHANGE DISTRIBUTED APPLICATION MODEL

The MIDA Standards must relate to the necessary formats and protocols which must exist between separate message systems to allow the open interchange of messages. Interfaces which exist between layers within a single message handling system are specifically excluded.

The areas addressed by this MIDA Standard are:

- Message Transfer Service description, Service Primitives and Protocol specification (P1).
- User Agent Service description and Protocol specification (P2).
- Naming and Addressing.
- Reliable Transfer Service and Protocol definition.

2.1 MIDA Architecture

2.1.1 General

In order to discuss the elements of standardization required for MIDA, it is necessary to formulate and adopt a viable model for a Message Handling system, which can be validated against current and future practical message passing applications packages, and which is aligned to the OSI Reference model.

- The Distributed Application for Message Interchange is within the Application Layer of the OSI Reference Model. This means that it would make use of the lower layer service (see clause 2.3).
- The Application Layer contains other functional entities, and it is necessary for the definition of a message handling system model, that the relationship of the message handling functionality with other application layer entities be established.
- The message handling functionality itself is divided into three sublayers. The upper sublayer is concerned with user visible features of the Message Handling System such as recipient to originator acknowledgment. This is termed the User Agent Sublayer (UAS). The middle sublayer is concerned with message passing from point to point between interconnected systems. This is termed the Message Transfer Sublayer (MTS).The lowest sublayer is a Reliable Transfer Service. This is the Application Layer functionality which undertakes the reliable transfer of one or more messages between Message Handling Systems.

The interconnection between Message Handling Systems has been defined as the "Communication Service" and this represents Layers 1-6 of the OSI Reference Model.

To complete the Application Layer, a "User Environment" is included above the UAS. This is outside the domain of the MIDA standardization, and relates to the facilities provided to users of the MIDA system.

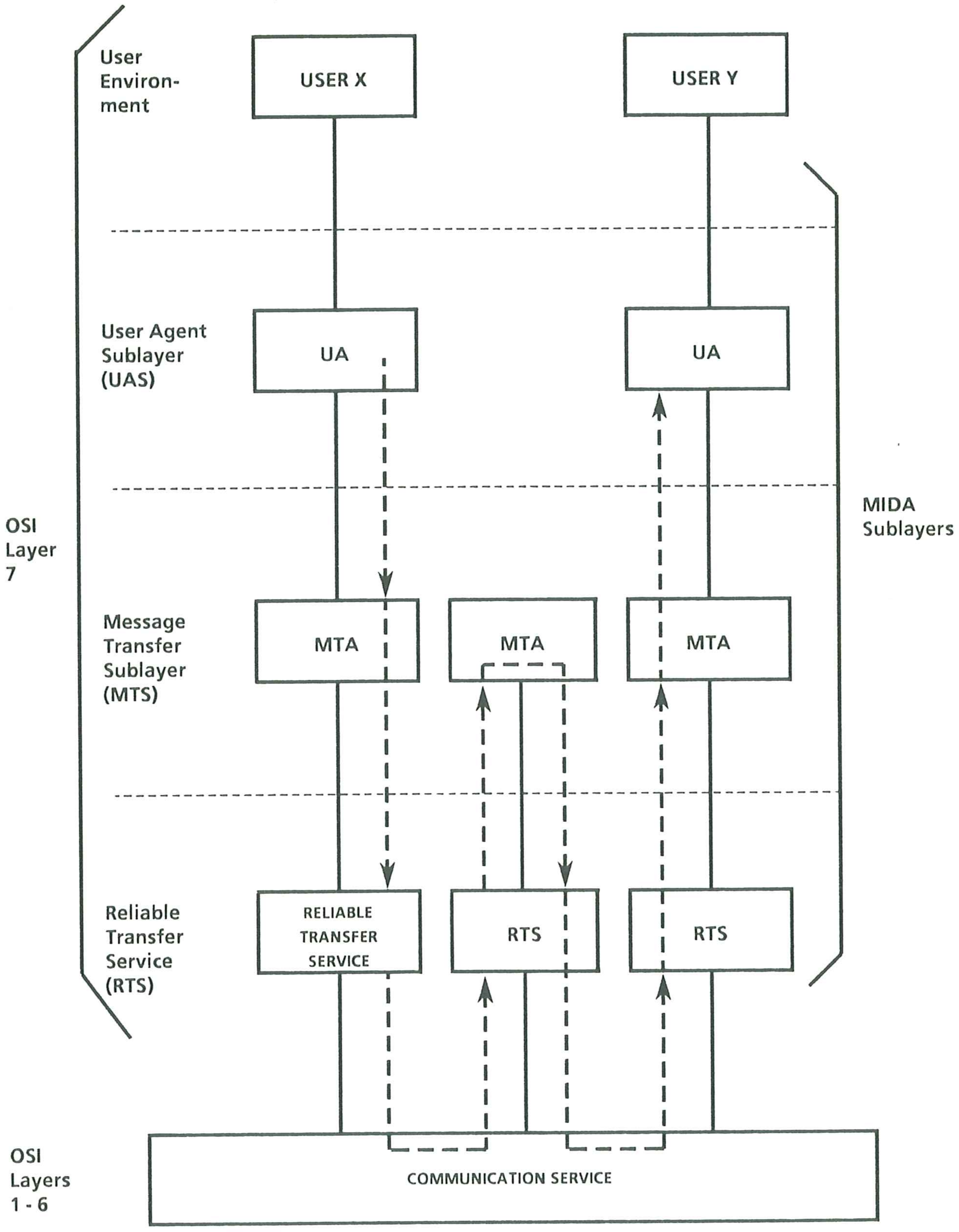

User Environment
USER X
USER Y
User Agent Sublayer (UAS)
UA
UA
OSI Layer 7
MIDA Sublayers
Message Transfer Sublayer (MTS)
MTA
MTA
MTA
Reliable Transfer Service (RTS)
RELIABLE TRANSFER SERVICE
RTS
RTS
OSI Layers 1 - 6
COMMUNICATION SERVICE

Figure 1. Model of the Message Handling System

2.1.2 Layers in the MIDA Model

2.1.2.1 User Environment Sublayer

Users of the Message Handling System may have facilities provided to them in the user environment sublayer which is the highest sublayer of layer 7 of the OSI Reference Model. The user is provided with facilities to allow him to create documents or other information packages for transfer through the Message Handling System. The nature of these facilities, and those by which the user may examine messages passed to him, are not the subject of MIDA standards.

The MIDA standards do identify the message handling control information which the user must supply to the Message Handling System for the correct forwarding of his message, see "User Agent Sublayer Services". This control information includes mandatory information, which must be supplied in all cases, and optional information, which may be supplied or omitted. However, if optional information is supplied, the nature of that information must be in accord with MIDA standards to allow correct analysis and action throughout the interconnected Message Handling Systems.

2.1.2.2 User Agent Sublayer

The User Agent Sublayer is the upper of the three sublayers into which the Message Handling functionality has been divided for the purposes of this Standard. It contains entities called "User Agents" which act on behalf of the users of the Message Handling System, and represent the user to the system. It also gains access to service entities such as Directory Services with which the user may wish to communicate. It contains the "User visible" constituents of the Message Handling System.

2.1.2.3 Message Transfer Sublayer

The Message Transfer Sublayer is the middle of the three sublayers of the Message Handling Functionality of the Message Handling System model. It contains entities known as "Message Transfer Agents" (MTA) which are responsible for determining the correct routing for a message to reach its destination or destinations, and for passing the message from one to another until it emerges into the User Agent sublayer at each required destination.

2.1.2.4 Reliable Transfer Service

This application sublayer is the lowest of the three sublayers. It provides the reliable transfer of one or more messages from the "sending" MTA, into the "receiving" MTA.

2.1.2.5 Communication Service

The Communication Service is the underlying mechanism which the Reliable Transfer Service utilises. It corresponds to layers 1-6 of the OSI Reference Model.

2.1.3 Other MIDA Features

2.1.3.1 Naming and Addressing

To enable the correct forwarding and delivery of messages it will be necessary to define a mechanism for the unique and unambiguous identification of all MTAs and the associated User Agents.

2.1.3.2 Message Format Standard

The Message format requirements from a semantic viewpoint are dictated by the earlier aspects of standardization. As exchanged between MTA's a message will comprise three sets of subfields listed below:

- Message Transfer Protocol Features;
- User Agent Protocol Features;
- User Information to be transferred.

The Message Format standard must define an encoding mechanism such that the various useful fields can be readily and unambiguously identified.

2.1.3.3 Organizational Mapping

This subsection defines the various roles that an organization may play in using MIDA services. An organization may be a company or a non-commercial organization.

The collection consisting of at least one MIDA MTA and zero or more MIDA UAs owned by an organization (not being an Administration) is called a Private Management Domain, (PRMD).

The collection of at least one MTA and zero or more UAs owned by an administration and providing services as defined in CCITT Rec. X.400 is called an Administration Management Domain, (ADMD).

Figure 2 illustrates a number of technically feasible interconnections. However it should be noted that some of these connections may be subject to regulations in one or both countries.

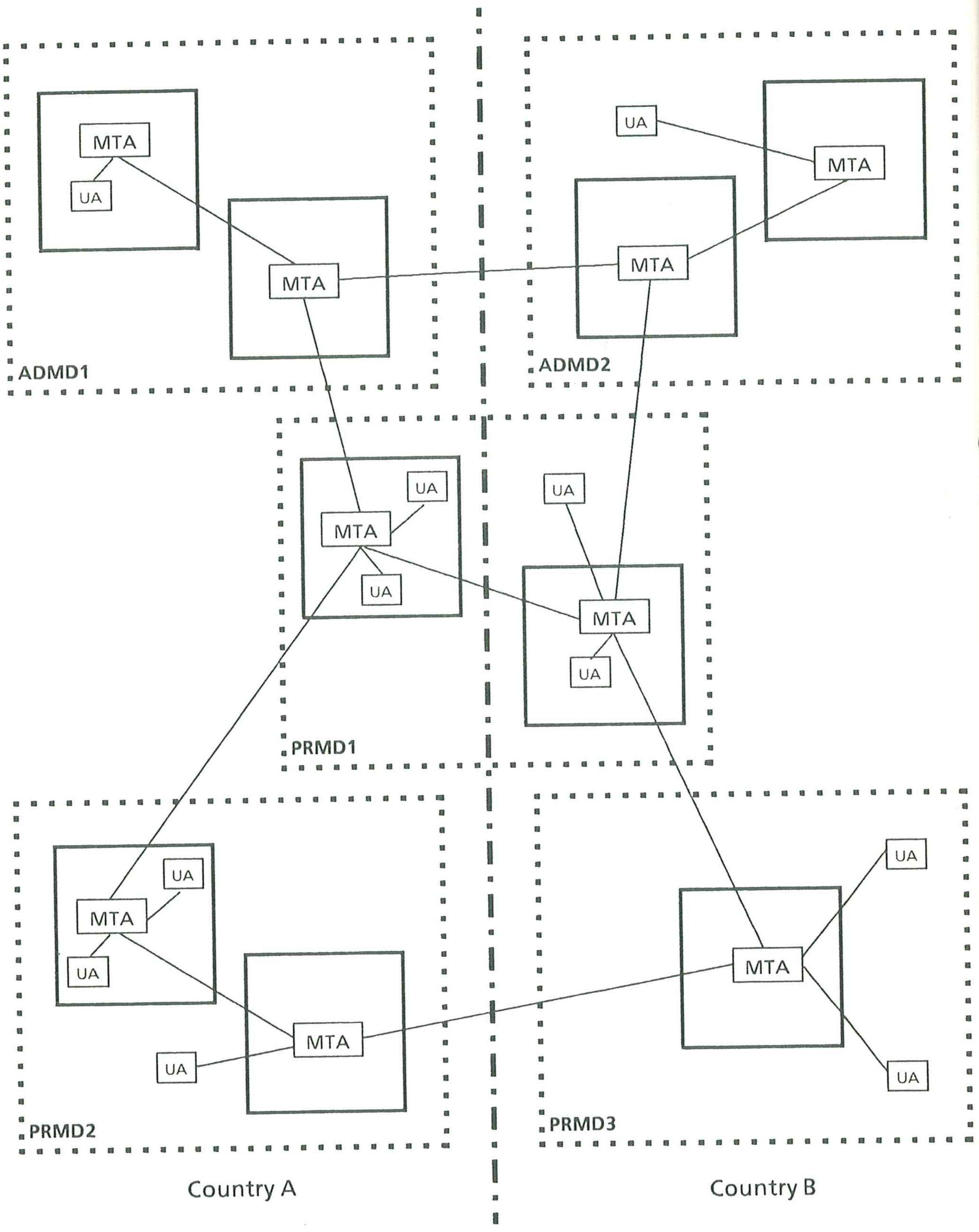


Figure 2. Inter-domain Relationship

2.2 MIDA Objects

The main objects handled by MIDA are shown in Figure 3 and Figure 4.

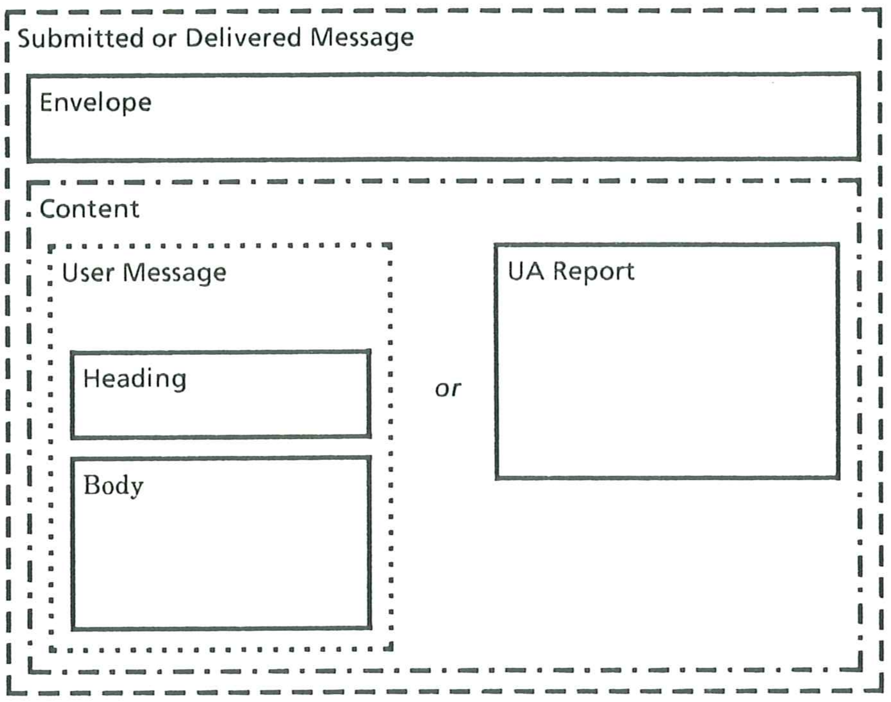


Figure 3. Logical view of MIDA objects

Heading	:	User Agent Protocol Control information
Envelope	:	Message Transfer Protocol Control Information
Content	:	It may be either a User Message or a User Agent report. It is also named Message content.

A User Message names the result of user information (Body) and UA services exchanged at the User-UA boundary.

A Message names the result of UA information (User Message or UA Report) and MTS services exchanged at the UA-MTS boundary.

A Body is the information the user wishes to communicate. The Body may consist of one or several body parts.

2.3 Services Required From Lower Layers

The MIDA model in Figure 1 of this Standard describes the Application Layer entities of the Message Handling System.

The Reliable Transfer Service of the MIDA model provides two alternative mappings onto the services provided by lower layers of the OSI Reference Model.

The Reliable Transfer Service operates in a fail safe manner ; any failure of the lower layer services, will result in it retrying the transfer until success is indicated, or a time out occurs. These repetitive attempts to such a successful transfer can, under certain failure conditions, cause a transfer to be completed twice.

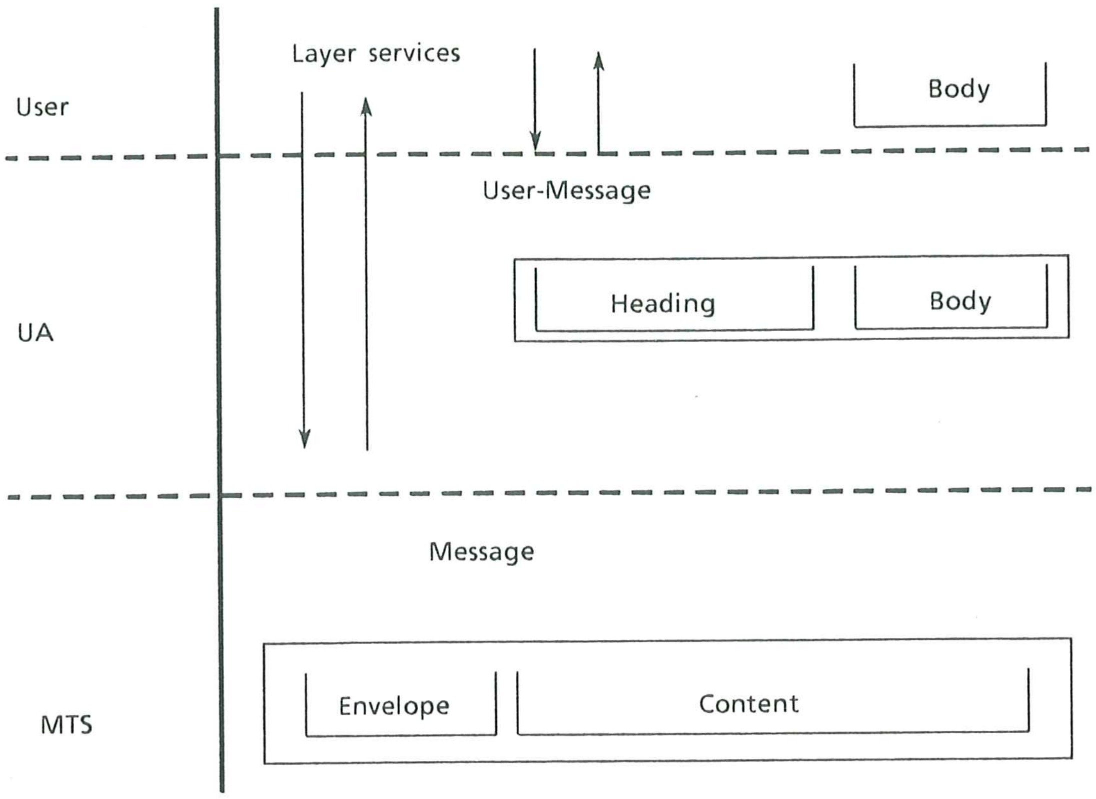


Figure 4. Layered view of MIDA objects

It is the responsibility of the recipient User Agent to ensure such duplicate messages are discarded by reference to the Message Identification in each received Message.

2.4 Naming and Addressing

The intent of this subsection is to explain the concept of originator/recipient (O/R) name and present the selection of O/R name attributes and forms.

2.4.1 Concepts and Terms

Two kinds of names: primitive and descriptive, are identified in the discussion of MIDA.

2.4.1.1 Primitive Names

A primitive name is a name assigned by a naming authority to a specific entity. A naming authority is simply a source of names. The only constraint imposed upon the naming authority is that it may never hand out the same name twice. For example, an employee number is a primitive name where the employing organization is the naming authority.

2.4.1.2 Descriptive Names

A descriptive name denotes exactly one user in MIDA.

If there is no unique user (none or several), its description (attributes) is not a descriptive name. However, if there is exactly one such user, the description denotes precisely that user and, therefore, is a descriptive name for him.

Primitive names typically figure as components of descriptive names. Since primitive names are unambiguous only within a limited context (that is, that of the naming authority), a single primitive name is in general not sufficient for globally unique identification.

2.4.1.3 Attributes and Attribute Lists

A descriptive name identifies an entity by specifying one or more of its attributes. The set of attributes and their associated values form an attribute list.

2.4.1.4 Originator/Recipient Names

In the MIDA context, the principal entity that requires naming is the user (the originator and recipient of messages). However , users are outside MIDA and therefore a descriptive name of a user is applied to the user's UA. This is called the O/R name.

To submit a message for delivery, the originating UA must provide the MTS with the O/R name of the UA for each of the message's intended recipients.

For the present, an O/R name is restricted to be an attribute list.

2.4.1.5 Originator/Recipient Addresses

Addresses are generally used to specify the geographical locations of buildings, the logical locations of components of information processing systems, and so forth. That is, an address usually identifies an entity by specifying where it is, rather than what it is.

An O/R address is a descriptive name for a UA that has certain characteristics that help the MTS to locate the UA's point of attachment.

Since an O/R address is also an O/R name, it could be used, for example, by the originator to identify the intended recipient of a message. One of the forms that an O/R Address may take is based on Architectual Characteristics of the Message Handling System. This O/R address is the only form of addressing that a MIDA system is required to support, to locate users. Its construction is:

- Country Code
- Administration Management Domain Name
- Private Management Domain Name
- MTA Name
- UA Local Identifier

It should be noted that the Management Domain Names identified above, require a Naming Authority to avoid duplication. This Standard assumes that such authorities will be created on a national or international basis.

2.4.2 Originator/Recipient Attributes

2.4.2.1 Construction of O/R Names

It is an objective that an originator be able to provide a descriptive name for each recipient of a message using information commonly known about that user. This Standard specifies a set of standard attributes from which these O/R names can be constructed.

Each Management Domain (MD) must ensure that every UA in the MD has at least one name. An MD does not necessarily have to utilise all attribute types in its MD when creating O/R names. However, it may allow its users to construct names using attributes utilised by other MDs.

Four categories of standard attributes are defined in MIDA. These categories, along with examples of possible attributes in each category, are given below. Those attributes marked with an asterisk (*) are CCITT-defined attributes and are encoded in the CCITT protocols.

I) Personal Attributes

Examples: Personal name*
(Surname, Given name(s), Initials, Generational qualifier (for example, "Jr.") , and Title (for example, "Dr."))

II) Geographical Attributes

Examples: Street name and number
Town name
Region name
Country name*

III) Organizational Attributes

Examples: Organization name*
Organizational unit*
Position or role

IV) Architectural Attributes

Examples: X.121 address*
Unique UA identifier (only numeric values)*
MTA name
UA local identifier
Administration Management Domain name *
Private Management Domain name*

NOTE 1
The MTA name and UA local identifier are carried as part of the Domain-Defined Attributes.

3. SERVICES OFFERED BY MIDA

This section describes the services and service primitives associated with the two sublayers of the MIDA model.

These services are classified as :

- Basic
- Optional

The definitions of these terms are :

Basic

A service the provision of which is mandatory in any system claiming conformance to this Standard. The associated protocol elements and encoding as defined herein must be supported (but note, a user may choose not to invoke all the service features offered as basic).

Optional

A service the protocol support of which is not mandatory. If the service is supported, however, it shall obey the protocol elements and encoding as defined in this Standard. An optional service should not be used in a message interchange without prior knowledge that all sending and receiving parties involved in such an interchange support it. A relaying party which does not support the optional service should nevertheless relay the associated protocol elements. A user may choose not to invoke some or all service features offered as optional.

The service elements are grouped into different sets depending on their functions :

- Fundamental
- Submission and Delivery
- UA Information
- Query
- User Message Operations
- User Message Attributes
- Body Attributes
- User Message Sending and Receiving Attributes
- Miscellaneous Attributes.

The service elements in the Fundamental set are specific in that they are implicitly in the protocols of this Standard. They enable messages to be sent or received by a user or UA.

3.1 Message Transfer Sublayer Services

3.1.1 MTS Services Overview

This clause specifies the services provided from the Message Transfer Sublayer by describing the interaction between a MTS service provider and a MTS service user, i.e., the User Agent of the functional model.

However, it is recognized that the mechanisms for gaining access to the message transfer sublayer are also candidates for standardization. This service definition therefore includes some services which could be required by an access mechanism but which have no direct mapping to the MIDA protocols.

3.1.1.1 Fundamental

3.1.1.1.1 Message Transfer

This service provides an orginator UA with the capability to submit a message to the MTS for transfer and delivery to one or more recipient UAs capable of receiving it. The MTS completes delivery when the message is given to the recipients UA along with the originators O/R name.

3.1.1.2 Submission and Delivery

3.1.1.2.1 Message Identification (Basic)

This service element enables the MTS to provide a UA with a unique identifier for each message submitted to or delivered by the MTS. UAs and the MTS use this identifier to refer to a previously submitted message in connection with service elements such as Delivery and Non-Delivery Notification

3.1.1.2.2 Non-Delivery Notification (Basic)

This service element enables the MTS to notify an originating UA if a submitted message was not delivered to the specified recipient UA(s). The reason the message was not delivered is included as part of the notification. For example, the recipient UA may be unknown to the MTS.

In the case of a multi-destination message, a non-delivery notification may refer to any or all of the recipient UAs to which the message could not be delivered.

3.1.1.2.3 Deferred Delivery (Optional)

This service element enables an originating UA to instruct the MTS that a message being submitted should be delivered no sooner than a specified date and time. Delivery will take place as close to the date and time specified as possible, but not before. The date and time specified for deferred delivery is subject to a limit which is defined by the originator's management domain. In private management domains there is no guarantee that a deferred delivery will be enforced except by holding the message at the originator´s MTA.

3.1.1.2.4 Delivery Notification (Basic)

This service element enables an originating UA to request that an explicit notification be returned to the originating UA when a submitted message has been successfully delivered to a recipient UA. The notification is related to the submitted message by means of the message identifier and includes the date and time of delivery. In the case of a multi-destination message, a delivery notification may refer to any or all of the recipient UAs to which the message was delivered.

Delivery notification carries no implication that any UA or user action, such as examination of the message's content, has taken place.

3.1.1.2.5 Delivery Time Stamp Indication (Basic)

This service element enables the MTS to indicate to a recipient UA the date and time at which the MTS delivered a message.

3.1.1.2.6 Grade of Delivery Selection (Basic)

This service element enables an originating UA to request that transfer through the MTS be urgent or deferrable, rather than normal. The time periods defined for deferrable and urgent transfer are longer and shorter, respectively, than that defined for normal transfer..

3.1.1.2.7 Multi-destination Delivery (Basic)

This service element enables an originating UA to specify that a message being submitted is to be delivered to more than one recipient UA. Simultaneous delivery to all specified UAs is not implied by the service. The number of recipient UAs on a submitted multi-recipient message is unlimited.

3.1.1.2.8 Submission Time Stamp Indication (Basic)

This service element enables the MTS to indicate to an originating UA and the recipient UA the date and time at which a message was submitted to the MTS.

3.1.1.2.9 Prevention of Non-delivery Notification (Basic)

This service element enables an originating UA to instruct the MTS not to return a non-delivery notification to the originating UA in the event that the message being submitted is judged undeliverable. This service is provided on a per recipient basis.

3.1.1.2.10 Cancellation Time Selection (Optional)

This service element enables an originating UA to instruct the MTS that a message being submitted should not be delivered after a specific date and time. A non-delivery notification is generated when a message is cancelled.

3.1.1.2.11 Alternate Recipient Allowed (Optional)

This service element enables an originating UA to specify that the message being submitted may be delivered to an alternate recipient as described below.

A destination MD will interpret some or all of the user attributes in order to select a recipient UA. Two cases may be distinguished:

- All the attributes match precisely those of a subscriber UA. The message is delivered to that UA.

- Either insufficient attributes are supplied or those supplied match those of more than one subscriber UA. The message cannot be delivered to the originator's intended UA.

There is no guarantee that the receiving Management Domain will not make an alternative delivery regardless of the originator invoking the service.

When a message is delivered to an alternate UA that has been assigned to receive undeliverable messages, the alternate UA will be notified of the O/R name of the intended recipient as specified by the originator. Delivery to the alternate UA will be reported in a delivery notification if requested by the originator.

NOTE 2
The availability of this service element to the originator does not imply that any alternate recipient UA exists or has been assigned to receive the message.

NOTE 3
The alternate recipient UA must be located in the same management domain as the intended recipient.

3.1.1.2.12 Originator-requested Alternate Recipient (Optional)

This service element allows the UA submitting a message to specify one alternate recipient to which the MTS may deliver the message if delivery to the original recipient's UA is not possible.

NOTE 4
If this service and the Alternate Recipient Allowed service are both implemented in the receiving domain and invoked by the originator, this service takes priority over Alternate Recipient Allowed.

NOTE 5
The originator-requested alternate recipient UA must be located in the same management domain as the intended recipient.

This service is on a per recipient basis.

3.1.1.2.13 Disclosure of Other Recipients (Basic)

This service element enables an originating UA to instruct the MTS, when submitting a multi-recipient message, to disclose the O/R names of all other recipient(s) to each recipient UA upon delivery of the message. The O/R names disclosed are as supplied by the originating UA.

The absence of this service element does not guarantee that O/R names will not be disclosed.

3.1.1.2.14 Return of Contents (Optional)

This service element enables an originating UA to request that the content of a submitted message be returned with any non-delivery notification.

3.1.1.3 UA Information

3.1.1.3.1 Content Type Indication (Basic)

This service element enables an originating UA to indicate the content type for each submitted message. A recipient UA may have one or more content types delivered to it.

3.1.1.3.2 Original Encoded Information Types Indication (Basic)

This service element enables an originating UA to specify to the MTS the encoded information types of a message being submitted. When a message is delivered, it also indicates to the recipient UA the encoded information types of the message specified by the originating UA.

This version of the MIDA Standard does not support conversion.

3.1.1.3.3 Converted Indication (Basic)

This service element enables the MTS to indicate to a recipient UA that the MTS performed encoded information type conversion on a delivered message. The recipient UA is informed of the resulting types.

This version of the MIDA Standard does not support conversion.

3.1.1.3.4 Conversion Prohibition (Basic)

This service element enables an originating UA to instruct the MTS that encoded information type conversion(s) should not be performed for a particular submitted message.

This version of the MIDA Standard does not support conversion.

3.1.1.4 Query

3.1.1.4.1 Probe (Basic)

This service element enables a UA to establish before submission whether a particular message could be deliverable. The MTS provides the submission information and generates delivery and/or non-delivery notifications indicating whether a message with the same submission information could be delivered to the specified recipient UAs.

The Probe service element includes the capability of checking whether the message size, content type, and/or encoded information types would render it undeliverable. The significance of the result of a Probe depends upon the recipient UA(s) having registered with the MTS the encoded information types, content type and maximum message size that it can accept. This service element is subject to the same delivery time targets as the urgent class.

3.1.1.4.2 Audit Trail (Basic)

This service element causes the message envelope to be marked at each entity in, the MTS, with the entity name and time of arrival. This information is available to the recipient´s UA, and is also returned to the originators UA as part of a Delivery Report in the case of an undeliverable message.

This service facility may be used by the Message Transfer Sublayer internally to detect looping of messages.

This service element is provided on a per recipient basis.

SET	SERVICE ELEMENT	BASIC	OPTIONAL
FUNDAMENTAL	Message Transfer		
SUBMISSION AND DELIVERY	Message Identification	X	
	Non-delivery Notification	X	
	Deferred Delivery		X
	Delivery Notification	X	
	Delivery Time Stamp Indication	X	
	Grade of Delivery Selection	X	
	Multi-destination Delivery	X	
	Submission Time Stamp Indication	X	
	Prevention of Non-delivery Notification	X	
	Cancellation Time Selection		X
	Alternate Recipient Allowed		X
	Originator-requested Alternate Recipient		X
	Disclosure of Other Recipients	X	
	Return of Contents		X
UA INFORMATION	Content Type Indication	X	
	Original Encoded Information Types Indication	X	
	Converted Indication	X	
	Conversion Prohibition	X	
QUERY	Probe	X	
	Audit Trail	X	

Table 1. Summary of the :ervices of the Message Transfer Sublayer

3.1.2 MTS Service Primitives

This clause describes the MTS services visible to the user. These services are defined in terms of primitives and parameters which are passed across the UAS/MTS boundaries. Using these primitives and parameters MTS users select the message interchange service they require.

Refer to Appendix C for a description of the conventions used in the remainder of this clause.

Name of primitives	Facility
MT-SUBMIT .request .confirmation	Message Submission Service
MT-DELIVER .indication	Message Delivery Service
MT-NOTIFY .indication	Message Notification Service
MT-PROBE .request .confirmation	Probe Service

Table 2. MTS Service Primitives

3.1.2.1 MT-SUBMIT: Message Submission Service

A UA Entity (UAE) issues the MT-SUBMIT.request primitive to initiate the transfer of a message to one or more recipients.

Two events are associated with MT-SUBMIT, as illustrated in Figure 5.

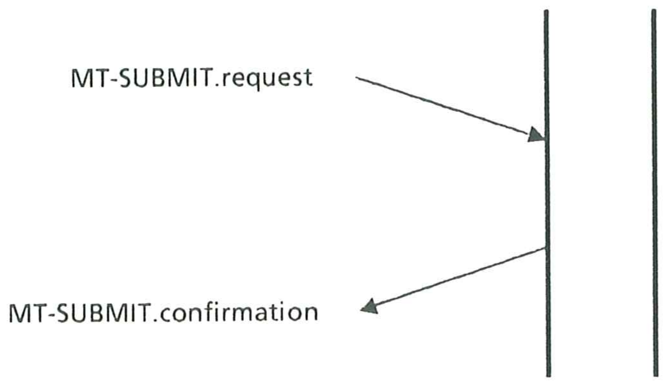


Figure 5. MT-SUBMIT Primitive Events

Parameter Name	Type	Parameter Description	Notes
Recipient-O/R-names	M	The O/R name(s) of the recipient(s)	6
Originator-O/R-name	M	The O/R name of the originator, which is given to the recipient when the message is delivered	6,7
Content	M	The information to be transferred	
Content-type	M	The content type of the message	10
Encoded-information-types	C	The type(s) of the information being transferred	8
NDN-suppress	C	Suppresses non-delivery notification	11,12
Priority	M	Selects the urgent, normal or deferrable delivery option	
Deferred-delivery-time	C	Specifies the date and time before which the message is not to be delivered	
Delivery-notice	C	Requests delivery notification	9,12
Conversion-prohibited	M	Requests that no conversion be performed	
Disclose-recipients	M	Indicates whether or not all recipient O/R names are to be revealed when the message is delivered	
Alternate-recipient-allowed	C	Indicates that the message may be delivered to an alternate recipient	
Content-return	C	Indicates that the content is to be returned as part of any non-delivery notification	11
UA-content-id	C	An identifier for the content of the message generated by the UAE. It may be used by the UAE for correlation of notifications with the content submitted	
Cancellation Time	C	Specifies date and time after which a message is not to be delivered	
Originator-requested-recipient-specified	C	Specifies an alternate recipient to whom the message is to be delivered if the original recipient is not available	12

Table 3. Parameters of the MT-SUBMIT.request primitive

NOTE 6
See clause 2.4 for the structure of O/R names.

NOTE 7
The local MTA must verify that the originator-O/R-name is legitimate, that is, that it denotes the originating UA and can be used as a recipient-O/R-name to send a message to the originator. As a local implementation detail, the originating MTA Entity (MTAE) may supply this parameter's value.

NOTE 8
This parameter is provided for the purposes of detecting encoded information type incompatibilities.

NOTE 9
Non-delivery notification is automatically provided in the event that a message cannot be delivered to a recipient UA.

NOTE 10
In case the MIDA-UA is the user of the MTS, the content type will indicate P2.

NOTE 11
The originating MTAE must not allow the combined use of the content-return and NDN-Suppress services in the same MT-SUBMIT.request.

NOTE 12
If the message is to be transferred to more than one recipient UA, a different value for this parameter may be specified for each recipient.

Parameter Name	Type	Parameter Description	Notes
Success-indication	M	Indicates the success or failure of the MT-SUBMIT.request	13
Submission-time	C	The time at which the request was accepted	14
Submit-event-id	C	The MT-SUBMIT.request event identifier	14, 15
Failure-reason	C	The reason for rejection of the MT-SUBMIT.request	16
UA-content-id	C	An identifier for the content of the message generated by the UAE	

Table 4. Parameters of the MT-SUBMIT.confirmation primitive

NOTE 13
Success indicates only that the MTS has accepted the request to transfer the message. It does not imply that the transfer and delivery of the message have been completed. Failure indicates that the MTS is unable to take responsibility for transferring the message.

NOTE 14
This parameter is present only if the MT-SUBMIT.request succeeds.

NOTE 15
This identifier is unique and meaningful only at the service-access-point where the MT-SUBMIT.request was issued. It is the means by which the MTS and originating UAE refer at their interface to a previously submitted message (for purposes, for example, of indicating a delivery or non-delivery notification).

NOTE 16
This parameter is present only if the MT-SUBMIT.request fails. The possible reasons for which the request might be refused include:

I) The recipient-O/R-names or another parameter is improperly specified.
II) The MTS's resources are inadequate to handle the message.
III) The originator-O/R-name is invalid.
IV) A service element is not subscribed.
V) Violation of the Control restrictions imposed by the MTS.

3.1.2.2 MT-PROBE: Probe Service

A UAE issues the *MT-PROBE.request* primitive to supply submission parameters to the MTS and inquire whether a MT-SUBMIT.request with the same parameters would be expected to result in successful delivery. For each recipient specified in the Probe, a delivery or non-delivery notification is returned.

Two events are associated with MT-PROBE, as illustrated in Figure 6.

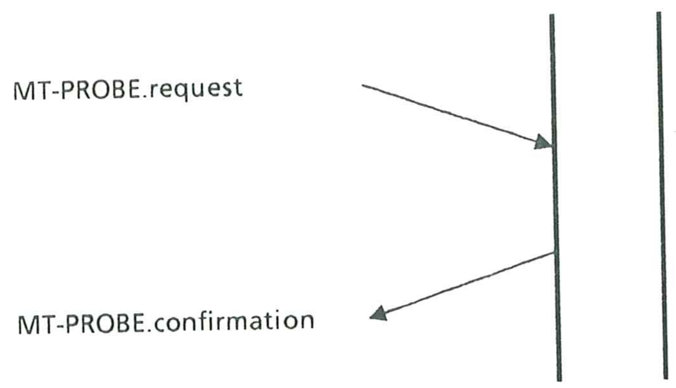


Figure 6. MT-PROBE Primitive Events

Parameter Name	Type	Parameter Description
Recipient-O/R-names	M	One or more recipient O/R names
Originator-O/R-name	M	The O/R name of the originator
Content-type	M	The content type of the message
Encoded-information-types	C	The encoded information type(s) of the message being probed
Conversion-prohibited	M	Requests that no conversion (of the message being probed) be performed
Alternate-recipient-allowed	C	Indicates that the Probe may test the possibility of delivery to an alternate recipient
Content-length	C	The estimated length in octets of the content of the message being probed
UA-content-id	C	An identifier for the content of the message generated by the UAE

Table 5. Parameters of the MT-PROBE.request primitive

Parameter Name	Type	Parameter Description	Notes
Success-indication	M	Indicates the success or failure of the MT-PROBE.request	
Probe-time	C	The time at which the request was accepted	17
Probe-event-id	C	The MT-PROBE.request event identifier	17
Failure-reason	C	The reason for rejection of the MT-PROBE.request	18
UA-content-id	C	An identifier for the content of the message generated by the UAE	17

Table 6. Parameters of the MT-PROBE.confirmation primitive

NOTE 17
This parameter is present only if the MT-PROBE.request succeeds.

NOTE 18
This parameter is present only if the MT-PROBE.request fails. The possible reasons for failure include the following:

i) The recipient-O/R-names or another parameter is improperly specified.
ii) A control has been violated.
iii) The originator-O/R-name is invalid.
iv) A service element is not subscribed.
v) Violation of the Control Restriction imposed by the MTS.

3.1.2.3 MT-DELIVER: Message Delivery Service

The MTS issues the MT-DELIVER.indication primitive to deliver a message to a recipient UAE. The recipient UA cannot refuse delivery of the message.

One event is associated with MT-DELIVER, as illustrated in Figure 7.

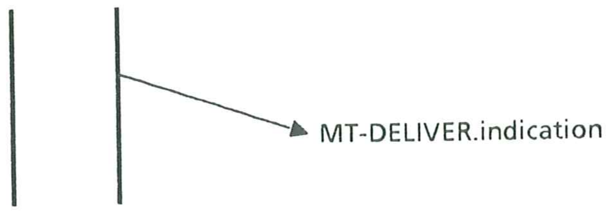


Figure 7. MT-DELIVER Primitive Event

Parameter Name	Type	Parameter Description	Notes
Originator-O/R-name	M	The O/R name of the originator	19
This-recipient-O/R-name	M	The O/R name of this recipient	
Other-recipient-O/R-names	C	The O/R name(s) of all other specified recipients of this message	20
Content	M	The information being delivered	
Content-type	M	The content type of the message	
Converted-encoded-information-types	C	The encoded information type(s) of the information being delivered, if different from Original-encoded-information types	
Original-encoded-information-types	C	The message's original encoded information type(s)	
Delivery-time	M	The time at which the message is being delivered	
Submission-time	M	The time at which the message was submitted	
Priority	M	Indication of urgent, normal, or deferrable delivery	
Intended-recipient. O/R-name	C	The O/R name of the intended recipient, as specified by the originator	21
Conversion-prohibited	C	Indication that conversion prohibition was requested by the originator	
Deliver-event-id	M	The MT-DELIVER. indication event identifier	

Table 7. Parameters of the MT-DELIVER.indication primitive

NOTE 19
This parameter is the O/R name that was supplied by the originating UAE when the message was submitted and verified to be correct by the originating MTAE.

NOTE 20
When Disclosure of Other Recipients is selected by the originating UAE, these are the other recipient O/R names specified.

NOTE 21
This parameter is present only if the message is being delivered to an alternate recipient.

3.1.2.4 MT-NOTIFY: Message Notification Service

The MTS issues the MT-NOTIFY.indication primitive to inform the UA that a previously submitted message was delivered or could not be delivered, or to convey the result of a Probe. The submission of a message or Probe addressed to two or more recipients may provoke several occurrences of this primitive.

One event is associated with MT-NOTIFY, as illustrated in Figure 8.

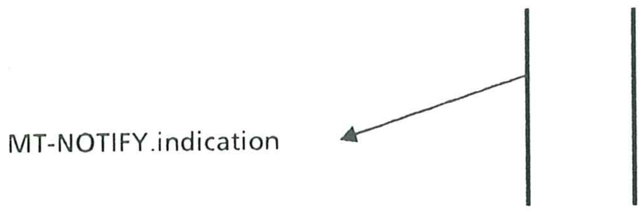


Figure 8. MT-NOTIFY Primitive Event

Parameter Name	Type	Parameter Description	Notes
Notification-type	M	Indicates whether the notification is of delivery or non-delivery	28
Submit-or-Probe-event-id	M	Identifies the MT-SUBMIT.request or MT-PROBE.request event associated with the message or probe to which the notification refers	22
Recipient-O/R-names	M	The O/R name(s) of the original recipient(s) to whom the notification applies	
Non-delivery-reason	C	The reason for non-delivery	23, 24, 28
Delivery-time	C	The time at which the message was actually delivered	25, 28
Converted-encoded-info-types	C	If conversion occurred (outside MIDA), the resulting encoded information type(s)	26, 27, 28
Intended-recipient. O/R-name	C	The O/R name of the original recipient as specified by the originator, in case the failure occurred while attempting to deliver to an alternate recipient or the message was delivered to an alternate recipient	28
Supplementary-information	C	Additional information for possible use by telematic services. Not supported by MIDA	
Returned-Content	C	For non-delivery, the content of the message, if content return was requested under these circumstances	
Type-of-UA	C	For delivery, this parameter indicates whether the recipient UAE was privately owned, or owned by an Administration	28
UA-content-id	C	An identifier for the content of the message generated by the UAE	

Table 8. Parameters of the MT-NOTIFY.indication primitive

NOTE 22
The event-id corresponds to that supplied by the MTS in the MT-SUBMIT.confirmation or the MT-PROBE.confirmation primitive when the message was originally submitted and accepted.

NOTE 23
The possible reasons for non-delivery include the following:
- *The O/R name is unrecognized.*
- *The O/R name is ambiguous.*
- *The message could not be transferred between MDs, within a particular MD, or from the MTS to the recipient UAE.*
- *The message's encoded information types are unsupported by the recipient UAE.*

- *The maximum time for delivering the message expired. The UAE is not accepting messages (in case of Hold for Delivery, ECMA-zz)*
- *Cancellation time reached.*

NOTE 24
This parameter is present only if a non-delivery or probe failure is being reported.

NOTE 25
This parameter is present only if a successful delivery is being reported.

NOTE 26
This parameter is absent from the response to a probe.

NOTE 27
This parameter and the conversion indicated apply to all of the recipient-O/R-names.

NOTE 28
If the notification signals non-delivery to or probe failure for more than one recipient UAE, a different value for this parameter may be specified for each recipient.

3.2 User Agent Sublayer Services

3.2.1 UAS Service Overview

The User Agent Sublayer makes available all the services from the MTS associated with:

- Fundamental,
- Submission and Delivery,
- UA information,
- Query,

exactly as described in clause 3.1.

These services are extended in this clause to include the services which are unique to the User Agent Sublayer. The categorization Basic is subdivided as follows for these extended services:

- Basic Receiving (BR);	Must be supported by a receiving UA
- Basic Sending (BS) ;	Must be supported by a sending UA
- Basic Sending and Receiving (BSR).	Must be supported by both sending and receiving UAs.

3.2.1.1 Fundamental

3.2.1.1.1 User Message Receipt

This service element enables the user to receive a delivered User Message.

3.2.1.1.2 User Message Sending

This service element enables the user to send a User Message to one or more recipients. This service does not imply simultaneous delivery to one or more recipients.

3.2.1.1.3 Body

This is the user information to be transferred by a MIDA system.

3.2.1.2 User Message Operations

3.2.1.2.1 Auto-forwarded Indication (BR)

The Auto-forwarding service allows a user to designate another single user to whom his UA should forward all delivered messages for a period of time (with conditions to be specified). The delivered message is auto-forwarded as a bodypart of a new user message.

This service element allows a recipient to determine that a body of an incoming User Message contains a User Message that has been auto-forwarded. Thus the recipient can distinguish from that where an incoming User Message contains a forwarded User Message in the body. As with a forwarded User Message, an auto-forwarded

User Message may be accompanied by information (for example, time-stamps, indication of conversion) associated with its original delivery.

Note 24
The indication that auto-forwarding of a User Message has occurred enables a recipient UA, should it so choose, to prevent further auto-forwarding and thus the possibility of loops. In addition, a recipient UA can choose whether or not to auto-forward based on other criteria (for example, sensitivity classification).

When a UA auto-forwards a User Message, it designates it as auto-forwarded. If receipt notification has been requested for the User Message being auto-forwarded, the UA generates a non-receipt notification informing the originator of the auto-forwarding of the User Message. The notification optionally includes a comment supplied by the originally intended recipient. No further notification applying to the auto-forwarded User Message is generated by any UA.

3.2.1.2.2 Forwarded User Message Indication (BR)

The forwarding service allows a user to send a previously received User Message to new set of recipients.

This service element allows a delivered User Message, or a delivered User Message plus its Delivery Information, to be sent as the body (or as one of the body parts) of a User Message as shown in Figure 9. An indication that the body part is forwarded is conveyed along with the body part. In a multi-part body, forwarded body parts can be included along with body parts of other types. Delivery information is information which is conveyed from the MTS when a User Message is delivered (for example, time stamps and indication of conversion). However, inclusion of this delivery information along with a forwarded User Message in no way guarantees that this delivery information is validated by the MTS.

The Receipt and Non-receipt Notification service elements are not affected by the forwarding of a User Message.

3.2.1.2.3 Replying User Message Indication (BSR)

The message replying service allows a recipient of a User Message to send a reply.

This service element allows the originator of a User Message to indicate to the recipient(s) that this User Message is being sent in reply to another User Message. If, by means of the Reply Request Indication service element, the originator of the original message specified the intended recipients of the reply, the originator of the reply should address it to those users. Otherwise the reply should be sent only to the originator. The originator of the reply may also specify the O/R names of additional users who are to receive copies of the reply for information.

The recipients of the reply receive it as a regular User Message, together with an indication of to which User Message it is a reply.

3.2.1.3 User Message Attributes

3.2.1.3.1 User Message Identification (BSR)

This service element enables the user to provide its own private unique identifier with the User Message, or to request this identification to be supplied by the UA. However, it is important to be aware that UAs and users use this identifier to refer to a previously sent or received User Message (for example, in receipt notifications).

The User Message identifier can convey with it the O/R name of the originator (or originating UA) which generated it. It is suggested that the O/R name be included with the identifier under any circumstances where identifiers generated by multiple users are involved, in order to prevent ambiguity.

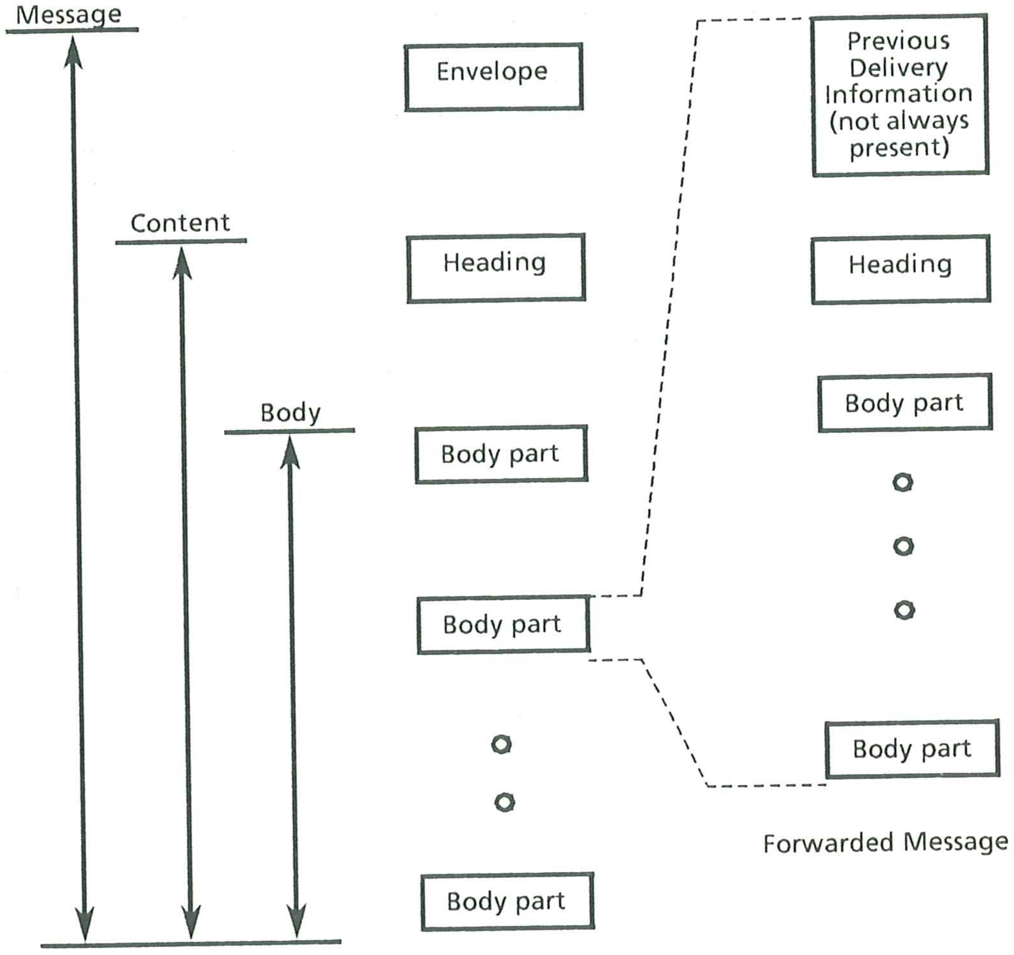


Figure 9. User Message

3.2.1.3.2 Cross-referencing Indication (BR)

This service element allows the originator to associate with the User Message being sent the identifiers of one or more other User Messages. This allows the recipient's UA, for example, to retrieve from storage a copy of the referenced User Messages.

3.2.1.3.3 Expiry Date Indication (BR)

This service element allows the originator to indicate to the recipient the date and time after which he considers the User Message to be invalid. The intent of this service element is to state the originator's assessment of the current applicability of a User Message. The particular action on behalf of a recipient by his UA, or by the recipient himself, is unspecified. Possible actions might be to file or delete the User Message after the expiry date has passed.

3.2.1.3.4 Importance Indication (BR)

This service element allows the originator to indicate to the recipients his assessment of the importance of the User Message being sent. Three levels of importance are defined: low, normal and high.

This service element is not related to the Grade of Delivery Selection service element provided by the MTS. The particular action taken by the recipient or his UA based on the importance categorization is unspecified. It is the intent to allow the recipient UA, for example, to present User Messages in order of their importance or to alert the recipient of the arrival of User Messages of high importance.

3.2.1.3.5 Obsoleting Indication (BR)

This service element allows the originator to indicate that one or more User Messages he sent previously are obsolete. The User Message that carries this indication supersedes the obsolete User message.

The action to be taken by the recipient or his UA is a local matter. The intent, however, is to allow the UA or the recipient to, for example, remove or file obsolete User Messages.

3.2.1.3.6 Sensitivity Indication (BR)

This service element allows the originator of a User Message to specify guidelines for the relative security of the User Message upon its receipt. The action to be taken by the recipient UA is to be specified. The intent is that the sensitivity classification should control such matters as:

- Whether the User Message may be auto-forwarded.
- Whether the recipient must prove his identity to receive the User Message.
- Whether the User Message may be printed on a shared printer as the means of receipt.

Three specific levels of sensitivity above the default are defined:

Personal
The User Message is sent to the recipient as an individual, rather than to him in his role. There is no implication that the User Message is private, however.

Private
The User Message contains information that should be seen (or heard) only by the recipient and not by any one else. The recipient's UA can provide services to enforce this intent on behalf of the User Message's originator.

Company-confidential
The User Message contains information that should be handled according to company-specific procedures.

Note 30
Other levels of sensitivity indication are for further study.

3.2.1.3.7 Subject Indication (BSR)

This service element enables the originator to indicate to the recipient(s) the subject of the User Message being sent. The subject information is to be made available to the recipient.

3.2.1.3.8 Multi-part Body (BR)

This service element allows an originator to send to a recipient or recipients a User Message with a body that is partitioned into several parts. The nature and attributes, or type, of each body part is conveyed along with the body part.

3.2.1.4 Body Attributes

3.2.1.4.1 Body Part Encryption Indication (BR)

This service element allows the originator to indicate to the recipient that any body part of the User Message being sent that has been encrypted. Encryption can be used to prevent unauthorized inspection or modification of the body part. This service element can be used by the recipient to determine that some body part(s) of the User Message must be decrypted. However, this service element does not itself encrypt or decrypt any body part.

3.2.1.4.2 Typed Body Part (BSR)

This service element allows the originator to identify the structural nature of the body part of the User Message. An ECMA-defined, ISO defined, nationally defined, or private-use type may be designated. Each recognized type governs the format ;constituent parts, if any (for example, mixed text and illustrations) ; and User Environment processing upon creation by, or presentation to, a user. A recognized type may have user-selectable options.The Body Part types which are defined in this version of the Standard are :

- All types defined by CCITT Rec. X.420;

in addition to these:

- ODIF; See ECMA-xx
- Transparent (Octet String);
- ISO 6937 Text.

3.2.1.5 User Message Sending and Receiving Attributes

3.2.1.5.1 Primary and Copy Recipients Indication (BSR)

This service element allows the originator to provide the names of the one or more users who are the intended primary recipients of the User Message, and the names of the zero or more users who are the intended copy recipients of the User Message. It is intended to enable a recipient to determine the category in which each of the specified recipients (including the recipient himself) was placed. The exact distinction between these two categories of recipients is unspecified. However, the primary recipients, for example, might be expected to act upon the User Message, while the copy recipients might be sent the User Message for information only.

Note 31
As an example of this service element in a typical memorandum, the primary

recipients are normally designated by the directive "to:" while "cc:" identifies the copy recipients.

3.2.1.5.2 <u>Blind Copy Recipient Indication (BR)</u>

This service element allows the originator to provide the names of one or more additional users who are intended recipients of the User Message being sent. It informs the recipient that he is receiving the message as a Blind Carbon Copy, i.e. that the originator did not wish to disclose this recipient to the primary and secondary recipients of the message.

3.2.1.5.3 <u>Receipt Notification (optional)</u>

This service element allows the originator to request that he be notified of the receipt, by a recipient, of a User Message being sent. The service element also implicitly requests notification of non-receipt as defined in clause 3.2.1.5.4.

The recipient's UA can return the receipt notification automatically as, for example, in the case where the User Message is automatically printed on a terminal. Alternatively, the recipient's UA can require positive action from the recipient to initiate the return of the notification. In this case, the recipient can elect not to have a receipt notification returned, even though the User Message has been received.

Besides the indication of receipt, the notification includes the following:

- The identifier of the User Message to which the notification applies.
- The time at which receipt occurred.
- Information describing the manner in which notification was initiated (for example, automatically by the UA or upon explicit action by the recipient).
- An indication of any conversion performed on the User Message.
- The O/R name of the recipient of the User Message and, in the event that the User Message was received by an alternate recipient's UA, the O/R name of the originally intended recipient as well.

<u>Note 32</u>
No legal significance can be attached to a receipt notification.

3.2.1.5.4 <u>Non-receipt Notification (Optional)</u>

This service element allows the originator to request that he be notified that a User Message was not received by the intended recipient. For multi-recipient User Messages, this service element can be specified on a per-recipient basis.

The recipient's UA is expected to return a non-receipt notification in the following circumstances:

- The User Message was auto-forwarded to another recipient.
- The recipient's subscription was terminated before receipt could occur.
- The User Message was discarded by the recipient's UA before receipt could take place.

Non-receipt notifications are generated automatically by a recipient's UA–the recipient himself is not involved in generating the notification.

It should be recognized that, in certain cases, a non-receipt notification will not occur even though the User Message has not been received. In particular, no time limit is used in defining when non-receipt has occurred. Consequently, the failure of the recipient to access his UA and receive his User Messages over a protracted period of time is not sufficient to cause a non-receipt notification to be generated.

Besides an indication of non-receipt, the notification includes the following:

- The identifier of the User Message to which the notification applies.

- The reason for non-receipt (for example, auto-forwarded, termination of subscription before receipt.)

- The O/R name of the recipient UA generating the notification, and in the event that the User Message was delivered to an alternate recipient UA, the O/R name of the originally intended recipient as well.

- An indication of any conversion performed on the User Message.

- Optionally, the return of the User Message, if requested by the originator. However the User message is not returned if conversion has taken place.

- In the case of auto-forwarding, optionally, a comment supplied by the intended recipient.

3.2.1.5.5 Reply Request Indication (BR)

This service element enables the originator to request that a recipient sends a User Message in reply to the User Message that carries the request. The originator can also specify the date by which the reply should be sent, and the O/R names of the one or more users to whom the reply should be sent. The recipient is informed of the date and names, but it is up to the recipient to decide whether or not to reply. A blind copy recipient should consider carefully to whom he sends a reply, in order that the meaning of the blind copy designation service element is preserved.

3.2.1.5.6 Authorizing Users Indication (BR)

This service element enables the originator to indicate to the recipient the names of the one or more persons who authorized its sending. For example, an individual may authorize a particular action which is subsequently communicated to those concerned by another person such as a secretary. The former person is said to authorize its sending while the latter person is the one who sent the message (originator). This does not imply signature-level authorization.

3.2.1.5.7 Originator Indication (BSR)

This service element allows the identity of the originator to be conveyed to the recipient. The MTS provides to the recipient the authenticated O/R name of the originator. In contrast the intent of this service element is to identify the originator in a user-friendly way.

3.2.1.6 Miscellaneous Attributes

3.2.1.6.1 Preparation Date (Optional)

This service element allows an originator to inform a recipient of the date on which a User Message was prepared.

3.2.1.6.2 Warning Date (Optional)

This service element allows a date prior to an expiring date to be defined.

SET	SERVICE ELEMENT	BASIC	OPTIONAL
FUNDAMENTAL	See Table 1.		
	User Message Receipt		
	User Message Sending		
	Body		
SUBMISSION AND DELIVERY	See Table 1.		
UA INFORMATION	See Table 1.		
QUERY	See Table 1.		
USER MESSAGE OPERATIONS	Auto-forwarded Indication	BR	
	Forwarded User Message Indication	BR	
	Replying User Message Indication	BSR	
USER MESSAGE ATTRIBUTES	User Message Identification	BSR	
	Cross-referencing Indication	BR	
	Expiry Date Indication	BR	
	Importance Indication	BR	
	Obsoleting Indication	BR	
	Sensitivity Indication	BR	
	Subject Indication	BSR	
	Multi-part Body	BR	
BODY ATTRIBUTES	Body Part Encryption Indication	BR	
	Typed Body Part	BSR	
USER MESSAGE SENDING AND RECEIVING ATTRIBUTES	Primary and Copy Recipient Indication	BSR	
	Blind Copy Recipient Indication	BR	
	Receipt Notification		X
	Non-receipt Notification		X
	Reply Request Indication	BR	
	Authorising Users Indication	BR	
	Originator Indication	BR	
MISCELLANEOUS ATTRIBUTES	Preparation Date		X
	Warning Date		X

Table 9. Summary of the Services of the User Agent Sublayer

3.3 Reliable Transfer Service

3.3.1 RTS Overview

The Reliable Transfer Service is used by the Message Transfer Agents to interchange messages.

The MTAs are not necessarily the only users of the RTS.

The mapping of the RTS onto the underlying layers can be realised in many ways, two of which are:

- using the Basic Activity Subset Session Services as specified in ISO 8326. This is the mapping in CCITT Rec. X.410 .
- using the Presentation Layer Services as specified in ECMA-84 and the Basic Syncronized Subset Services as specified in ISO 8326.

3.3.2 RTS Primitives

The interactions between the RTS and RTS-user are described as a set of service primitives. Service primitives are abstractions. They attempt to capture only those details of the interaction between entities that are aspects of the layer service itself. A service primitive neither specifies nor constrains the implementation of entities or the interfaces between them.

The service primitive descriptions that follow adhere to the conventions outlined in Appendix C.

3.3.2.1 RT-OPEN: Establishment of an Association

An RTS-user issues the RT-OPEN.request primitive to establish or open a new association with another RTS-user. The association may comprise several connections in sequence. These connections may be either session connections or presentation connections depending on the chosen mapping onto the underlaying layer.

Four events occur when an association is opened, as illustrated in Figure 10.

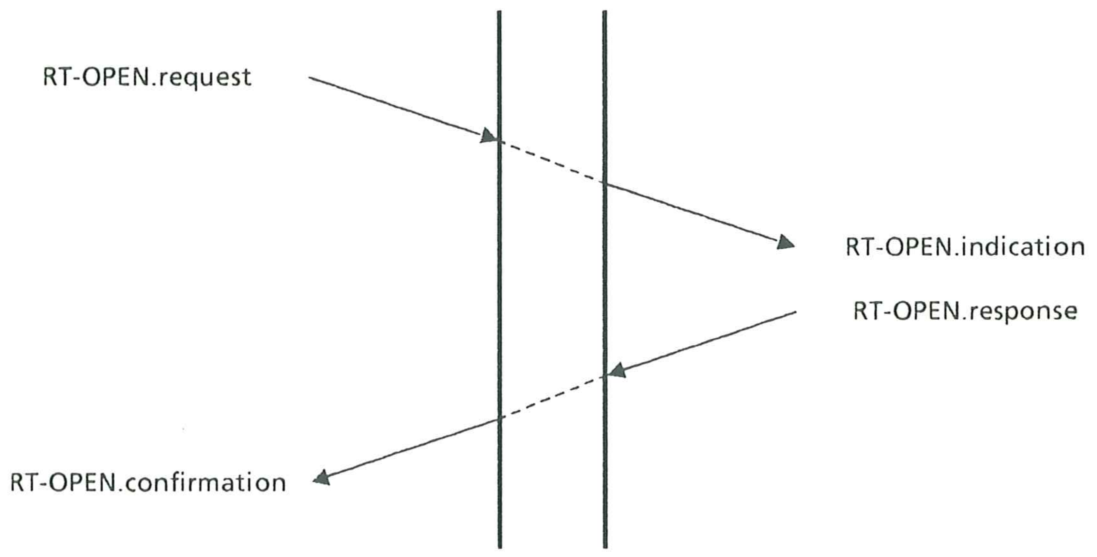


Figure 10. RT-OPEN Primitive Events

Parameter Name	Type	Parameter Description	Note
Responder-address	M	The address of the RTS associated with the RTS-user to which the new association is to be opened	
Dialogue-mode	M	The type of association to be opened: monologue or two-way alternate	
Initial-turn	M	The RTS-user that is to have the turn initially: initiator or responder	
Application-protocol	M	Designates the application protocol that will govern communication over the association	33
User-data	C	User data associated with opening the association	

Table 10. Parameters of the RT-OPEN.request primitive

Note 33
The value defined for this parameter is P1 for MIDA.

Parameter Name	Type	Parameter Description	Notes
Initiator-address	M	The address of the RTS associated with the RTS-user that issued the RT-OPEN.request primitive	
Dialogue-mode	M	The type of association being opened: monologue or two-way alternate	35
Initial-turn	M	The RTS-user that is to have the turn initially: initiator or responder	35
Application-protocol	M	Designates the application protocol that will govern communication over the association	34, 35
User-data	C	User data associated with opening the association	35

Table 11. Parameters of the RT-OPEN.indication primitive

Note 34
The value defined for this parameter is P1 for MIDA.

Note 35
This parameter conveys the same value as in the RT-OPEN.request primitive.

Parameter Name	Type	Parameter Description	Notes
Disposition	M	The disposition of the request for a new association: accepted or refused	
User-data	C	User data associated with accepting the association	36
Refusal-reason	C	The reason for refusing the association	37

Table 12. Parameters of the RT-OPEN.response primitive

Note 36
This parameter is present only if the association is accepted.

Note 37
This parameter is present only if the association is refused. Values defined are: unacceptable dialogue mode, authentication failure, busy.

The parameters of the RT-OPEN.confirmation primitive are identical to, and have the same values as, those of the RT-OPEN.response primitive.

3.3.2.2 RT-CLOSE: Release of an Association

The initiating RTS-user issues the RT-CLOSE.request primitive to release or close the association. It may do so only if it possesses the turn. If the turn does not exist, only the association initiator may initiate the RT-CLOSE.request.

Four events occur when an association is closed, as illustrated in Figure 11.

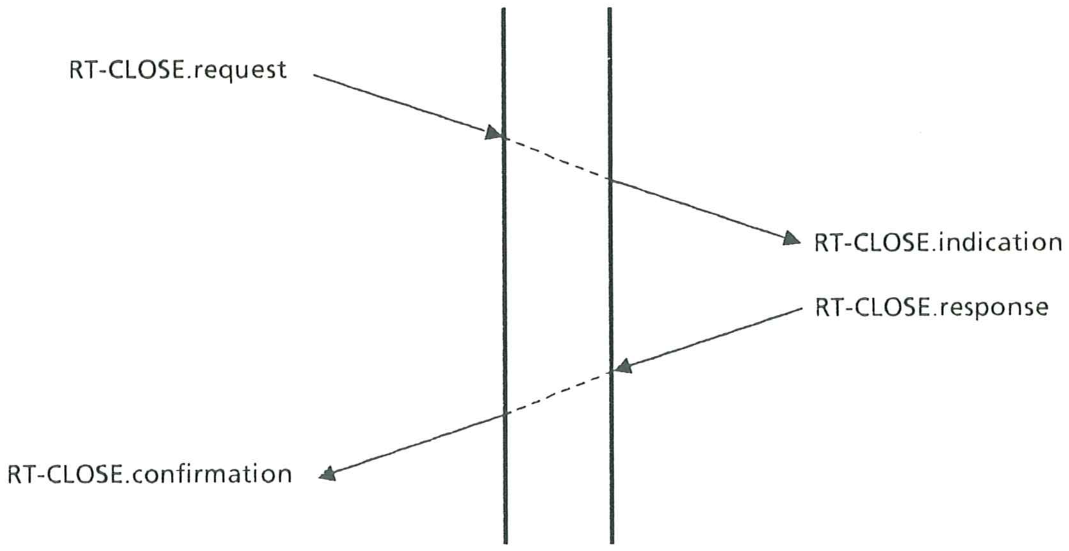


Figure 11. RT-CLOSE Primitive Events

None of the RT-CLOSE primitives has any parameters.

3.3.2.3 RT-TURN-PLEASE: Request for Exchange of the Turn
(Not available in monologue mode)

An RTS-user issues the RT-TURN-PLEASE.request primitive to request the turn. It may do so only if it does not already possess the turn. The turn is requested either to transfer data or release the association. The request conveys the priority of the action to be taken so that the other RTS-user can decide when to actually relinquish the turn.

Two events occur when exchange of the tokens is requested, as illustrated in Figure 12.

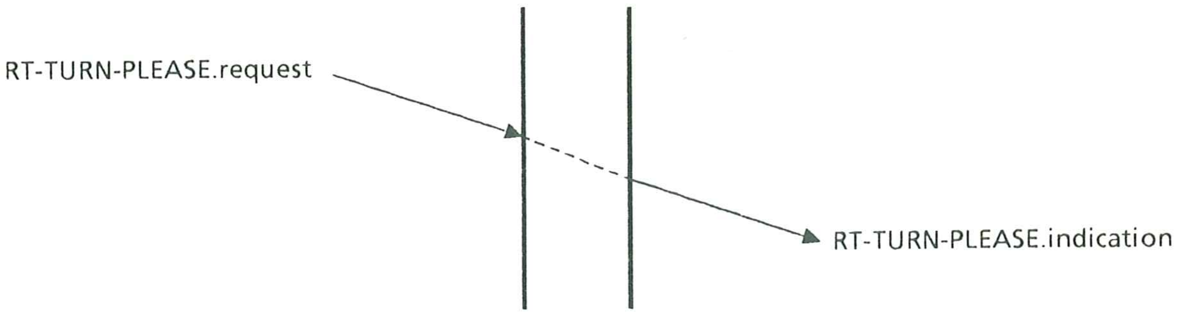


Figure 12. RT-TURN-PLEASE Primitive Events

Parameter Name	Type	Parameter Description	Note
Priority	C	The priority of the action, governed by the turn, that the requesting RTS-user wishes to carry out	38

Table 13. Parameters of the RT-TURN-PLEASE.request primitive

<u>Note 38</u>
A priority is assigned to each RTS-user action. The actions of releasing an association and of transferring various RSDUs will be assigned priorities. The range of valid priorities is a property of the application protocol in use.

The parameters of the RT-TURN-PLEASE.indication primitive are identical to, and have the same values as, those of the RT-TURN-PLEASE.request primitive.

3.3.2.4 RT-TURN-GIVE: Exchange of the Turn (Not available in monologue mode)

An RTS-user issues the RT-TURN-GIVE.request primitive to relinquish the turn to its peer. It may do so only if it possesses the turn.

Two events occur when the tokens are exchanged, as illustrated in Figure 13.

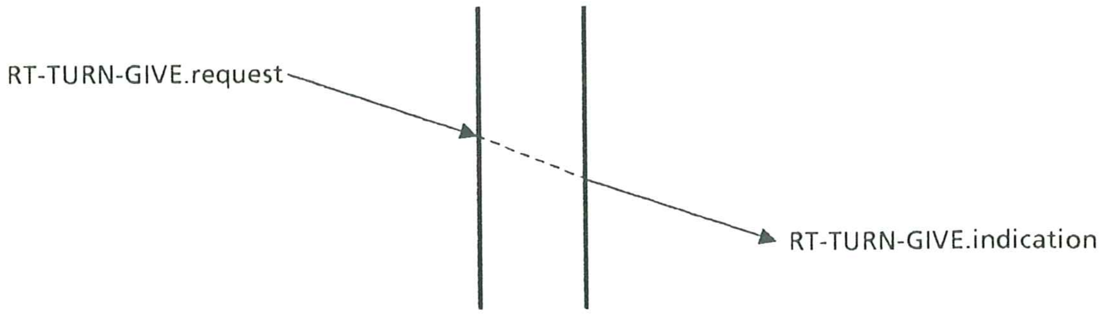


Figure 13. RT-TURN-GIVE Primitive Events

Neither the RT-TURN-GIVE.request nor RT-TURN-GIVE.indication primitive has any parameters.

3.3.2.5 RT-TRANSFER: Reliable Transfer of an Application Protocol Data Unit

An RTS-user issues the RT-TRANSFER.request primitive to request the reliable transfer of an RSDU over an association. It may do so only when the RTS is able to start sending data on an actual connection.

Two events occur when an RSDU is reliably transferred, as illustrated in Figure 14.

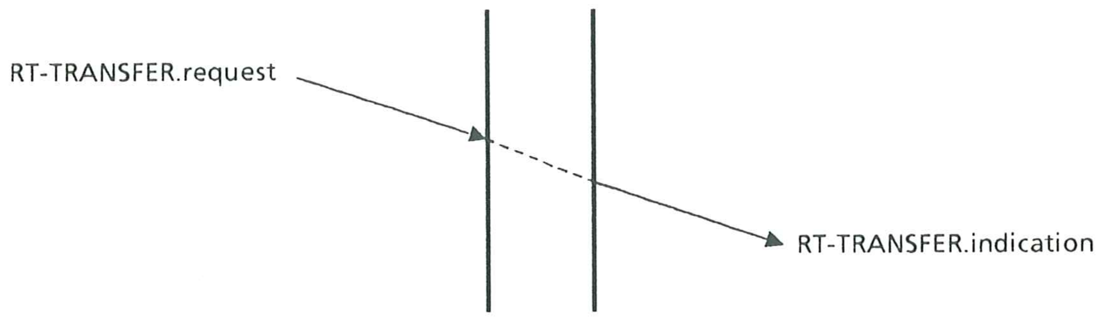


Figure 14. RT-TRANSFER Primitive Events

Parameter Name	Type	Parameter Description
RSDU	M	The RSDU to be transferred
Transfer-time	M	The time period within which the RTS must successfully transfer the RSDU to the other RTS-user

Table 14. Parameters of the RT-TRANSFER.request primitive

Parameter Name	Type	Parameter Description	Note
RSDU	M	The RSDU being transferred	39

Table 15. Parameters of the RT-TRANSFER.indication primitive

Note 39
This parameter conveys the same value as in the RT-TRANSFER.request primitive.

3.3.2.6 RT-EXCEPTION: Indication of Transfer Failure

The RTS issues the RT-EXCEPTION.indication primitive if it cannot complete the transfer of an RSDU as requested.

One event occurs when a transfer failure is indicated, as illustrated in Figure 15.

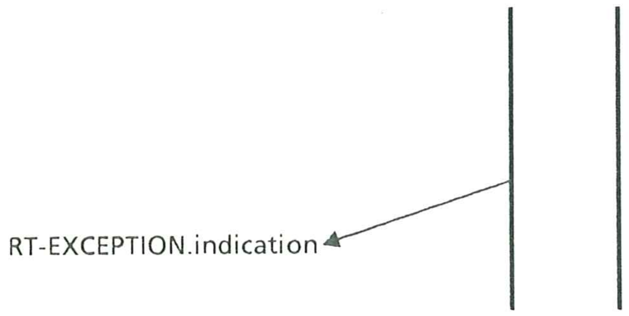


Figure 15. RT-EXCEPTION Primitive Event

Parameter Name	Type	Parameter Description
RSDU	M	The RSDU that could not be transferred within the allotted time

Table 16. Parameters of the RT-EXCEPTION.indication primitive

4. MIDA PROTOCOLS

4.1 Message Transfer Sublayer Protocol (P1)

4.1.1 Introduction

The MTS provides various services to the UAS, as specified in clause 3.1. Some of those services are provided by means of functions located in a single MTAE and thus do not depend upon communication between MTAEs. The provision of other services, however, requires that two or more MTAEs cooperate with one another. Such cooperation is achieved by means of the Message Transfer Protocol (P1).

The P1 protocol is used for communication between Message Transfer Agents.

The protocol elements of P1 are called Message Protocol Data Units (MPDUs). MPDUs are classified as either User MPDUs (UMPDUs) or Service MPDUs (SMPDUs), allowing easy identification of the functions for which cooperation is required. UMPDUs carry messages submitted by a UA for transfer and delivery to another UA. SMPDUs (for example, the Delivery Report MPDU) are used to convey information about messages between MTAEs.

The operation of P1 is such that MPDU parameters relating to optional user facilities not supported by a particular relaying MD do not cause non-delivery of the message, and are passed transparently through the MD; MPDU parameters not defined in this Standard, but encoded according to CCITT Rec. X.409, are passed transparently through a MIDA MTA.

4.1.2 Operation of the MTS

4.1.2.1 Routing

More than one MTAE may participate in the relaying and delivery of a message addressed to multiple recipients. If the various recipients are served by several different MTAEs, the message must be transferred through the MTS along several different paths, as illustrated in Figure 16. From the perspective of an MTAE relaying a message (for example, MTAE 1 in Figure 16, some recipients may be reached by one path while other recipients are reached by another. At such an MTAE, two copies of the MPDU are created, and each is relayed to the next MTAE along its respective path. The copying and branching of MPDUs is repeated until each copy has reached a final, destination MTAE, where the message can be delivered to one or more recipient UAs.

Every MTAE along the path taken by an MPDU is responsible for delivering or relaying the message to a particular subset of the originally specified recipients. Other MTAEs take care of the delivery or relaying to remaining recipients, possibly using copies of the MPDU created along the way.

A recipient information parameter containing a responsibility flag appears in the MPDU envelope for each originally specified recipient. Each relaying MTAE indicates to the next MTAE the recipients for which it is *not* to take responsibility in one of two ways:

- By setting their responsibility flags to false. If Disclosure of Other Recipients has been requested, this approach must be taken.

- By removing their recipient information parameters from the MPDU envelope.

The existence of the first alternative requires that MTAEs be able to process UMPDUs containing a mix of true and false responsibility flags, as well as UMPDUs containing only responsibility flags set to true.

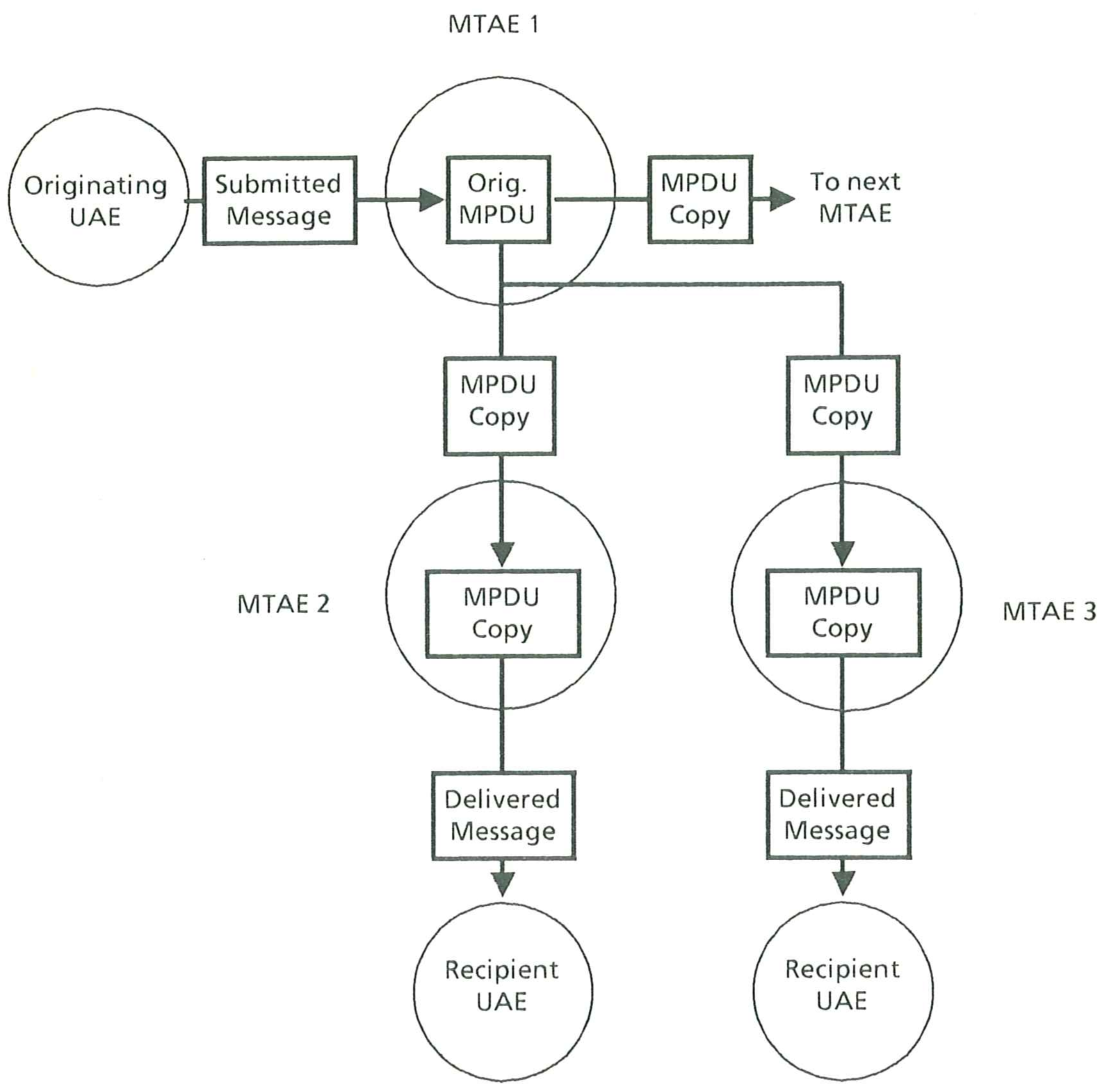


Figure 16. MTS Routing to Multiple Recipient UAEs

4.1.2.2 Delivery Reports

The MTS notifies UAs of the delivery or non-delivery of messages. The Message Transfer Protocol facilitates this by relaying SMPDUs called Delivery Report MPDUs between MTAEs. Delivery Report MPDUs are generated in accordance with the value of the Report Request parameters, which are additional pieces of recipient information. Reports provoked by the report request parameter may be for use within the MTS or may be used to notify a UAE of the delivery or non-delivery of a message. A single report may serve both purposes.

An MTAE generates a positive delivery report upon successfully delivering a copy of a message to the recipient UA. It generates a negative delivery report upon determining that a copy of a message is undeliverable to a particular recipient, that is, that it can take none of the following actions:

- Deliver the message to the recipient UA.

- Relay the message to an adjacent MTAE that would take responsibility for relaying it further.

- Redirect the message to an alternative recipient UA.

For efficiency, the MTAE may generate a single, combined delivery report that applies to several copies of a single multi-recipient message for which it is responsible. The Message Transfer Protocol makes allowance for this in the structure of the Delivery Report MPDU. However, in order for delivery reports to be combined in this manner, the same content conversion, if any, must have been performed on the message for all recipients to whom the delivery report refers.

Delivery reports that pertain to copies of the same multi-recipient message but that were generated by different MTAEs are not combined in the Message Transfer Protocol. Instead, they remain distinct and are transferred independently, even when routed through the same intermediate MTAE.

4.1.2.3 Content Conversion

Not supported by MIDA.

4.1.3 Operation of the MTAE

4.1.3.1 Introduction

This subsection describes the functions that an MTAE must perform, but does not dictate the MTAEs internal structure. Similarly, the functions performed when the originator and recipient are served by the same MTAE are not described. This description assumes that an RT-TRANSFER.request may be issued to the RTS at any time. The queuing that would have to exist in any real implementation is not described.

The MTAE is a logical machine that processes the Message Transfer Protocol. It is activated by either of the following two events at its interface with the UAS: an MT-SUBMIT.request or an MT-PROBE.request. It is also activated by either of two events at its interface with the RTS: an RT-TRANSFER.indication or an RT-EXCEPTION.indication.

Note 40
These primitives are assumed to be issued by a UAE co-located with the MTAE.

4.1.3.2 Action on MT-SUBMIT.request Event

I) The originator is validated (for example, by querying a local directory).

II) The parameters of the MT-SUBMIT.request are validated and a new UMPDU created. This validation is essentially syntactic; the semantics of the parameters are validated during the relaying of the MPDU.

III) The submit-event-id is generated.

IV) If the preceding actions are correctly carried out, an MT-SUBMIT.confirmation event indicating success are generated. Otherwise, the MT-SUBMIT.confirmation indicates failure.

V) If Deferred Delivery is requested either the message is held until the Deferred Delivery time has arrived, and then handled in the normal way; or, if a bilateral agreement exists, it may be relayed to another MTA with the Deferred Delivery time parameter included in the envelope.

Note 41
No procedure is defined to support the optional user facility of Deferred Message

Cancellation in the case where deferred delivery is supported by an MTAE other than the originating MTAE. Such a procedure is for further study.

VI) Each successfully generated UMPDU is passed to the RTS by means of an RT-TRANSFER. request.

4.1.3.3 Action on MT-PROBE.request Event

The actions taken are the same as those for an MT-SUBMIT.request, except that a Probe MPDU is created (unless there is a failure) and an MT-PROBE.confirmation event generated.

4.1.3.4 Action on RT-TRANSFER.indication Event

I) The MPDU envelope is validated. If the envelope is found invalid, a negative Delivery Report MPDU is returned, in the case of Probe and User MPDUs, provided that the originator O/R name is valid. The report carries a reason parameter indicating unable to transfer. An invalid MPDU is not processed further, but the event should be recorded so that diagnostic and corrective action can be taken.

II) If the message arrives from another MD, the global domain identifier of the MD and the arrival time are added to the trace information. The PerMTATraceInfo is added to the Internal Trace Info.

III) The MTA selects the recipients (i.e., O/R names to which the MPDU is addressed). for which the routing algorithm is to be executed. For Delivery Report MPDUs there is only one. For Probe and User MPDUs the recipients are those for which the responsibility flag is true. For each recipient, three results are possible as follows:

a) The MTA serves this recipient. The actions depend upon the MPDU type as follows:

<u>User MPDU</u>: If the message can be delivered, the delivery-stamp is generated and an MT-DELIVER.indication primitive is issued. If the message cannot be delivered (for example, because of encoded information type incompatibility) a Last Trace Information item is generated, incorporating the reason for non-delivery. If delivery is successful and if a confirmed or audit-and-confirmed report is requested, a Last Trace Information item indicating that delivery has been successfully completed is created. The MTAE may create a Delivery Report MPDU and generate an RT-TRANSFER.request. Alternatively, it may combine this with the Last Trace Information items for other recipients of the same message. At a later time, these will be incorporated in a Delivery Report MPDU, and an RT-TRANSFER.request will be generated.

<u>Delivery Report MPDU</u>: The message is correlated with the related submission event for local purposes (for example, charging, statistics, or trace). If the returned trace information contains a non-delivery indication, a (negative) MT-NOTIFY.indication event with the associated parameters is issued (unless prevention of Non Delivery Notification was requested at submission). If a delivery notification was requested and the returned trace information contains a delivery indication, a (positive) MT-NOTIFY.indication event is issued. If a copy of the MPDU was stored and all reports have been received, the MPDU is deleted. The time limit for storage when reports are not requested or do not come back is for further study.

<u>Probe MPDU</u>: If an MT-DELIVER.indication for a UMPDU with the same envelope parameters could be issued, the actions appropriate to completion of the MT-

DELIVER.indication are taken. Otherwise, the actions for an undeliverable UMPDU are taken.

For User and Delivery Report MPDUs, this description assumes that no control restrictions are in force that prevent the MPDU from being passed to the UA. If such restrictions exist, the MPDU must wait, and this may cause the delivery to be abandoned after a period of time. Messages that cannot be delivered because of registration restrictions can be declared undeliverable immediately.

b) The MPDU must be relayed to another MTAE. The action depends on MPDU type as follows:

<u>Probe or User MPDU</u>: The trace information is examined for loops. If a loop is detected, a Delivery Report MPDU with the appropriate trace information is generated. Otherwise, the Message Dispatcher proceeds as follows. If this is the first recipient for which the MPDU must be relayed to the particular other MTAE, then a copy of the MPDU is created, with the responsibility flags set to false for all of the other recipients. Otherwise, the responsibility flag for this recipient is set to true in the existing copy.

<u>Delivery Report MPDU</u>: The trace information is examined for loops. If a loop is detected, the MPDU is discarded. Otherwise the MPDU is relayed.

c) The routing algorithm indicates an error in the O/R name.

A Last Trace Information item indicating the error is generated. A Delivery Report MPDU may be generated or the trace information combined with other trace information.

IV) Operations when delivery to intended recipient is not possible and alternate recipient services are available.

Two mechanisms are defined, the Originator-requested Alternate Recipient, and the Alternate Recipient Allowed.

When an MTAE supports either or both alternate recipient mechanisms, it must perform an additional function to examine the Deliver To flag of each recipient for which it has routing responsibility, to determine whether the routing algorithm is to be executed with respect to the intended recipient, the originator-requested alternate recipient, or the (domain defined) alternate recipient.

When an MTAE has delivery responsiblity for a particular recipient, the impact of alternate recipient mechanisms in the case of failure are as follows :

a) The MTAE fails to make a delivery to the intended recipient. If an originator-requested alternate recipient is present, the MTAE sets the Deliver To flag to specify Deliver to originator-requested alternate recipient. The routing algorithm is reexecuted with respect to the originator-requested alternate recipient attributes. If there is no originator-requested alternate recipient, the MTAE may have knowledge of a domain defined alternate recipient, or "dead letter" position. The MTAE sets the Deliver To flag to specify Deliver To domain defined alternate recipient and reexecutes the routing algorithm with respect to that alternate recipient. If there is neither an originator-requested alternate recipient nor a domain defined alternate recipient the non-delivery mechanism is executed.

b) The MTAE fails to make a delivery to the orignator-requested alternate recipient.

If there is a domain defined alternate recipient, the MTAE sets the Deliver To flag to specify Deliver to domain defined alternate recipient, and reexecutes the routing algorithm with respect to that recipient. If there is no domain defined alternate recipient, the non-delivery mechanism is executed.

c) The MTAE fails to make a delivery to the domain defined alternate recipient. The non-delivery mechanism is executed.

It should be noted that whether an MTAE observes the Alternate Recipient Allowed parameter of the message which is non-deliverable, or not is a local matter of the receiving MTAE. An MTAE which executed rerouting of a message should update the last element of the Trace Information.

V) After executing the routing algorithm for each recipient, a check is made to ensure that all necessary Delivery Report MPDUs have been generated. An RT-TRANSFER.request is then issued for each MPDU to be relayed. The transfer-time parameter is set, taking into account the time the message has already spent in the MTS and (possibly) the availability of an alternate route. Before a User or Probe MPDU is relayed, recipient information parameters whose responsibility flags are false may be removed, unless Disclosure of Other Recipients was requested.

4.1.3.5 Action on RT-EXCEPTION.indication Event

I) If the MTA can perform alternate routing, the routing algorithm is re-executed for each recipient:

a) For each recipient for whom an alternate route is available, a determination is made whether a copy of the MPDU intended for other rerouted recipient(s) of that excepted MPDU has been prepared for the selected MTAE. If so, the copy´s responsibility flag for the current recipient is set to true. Otherwise a copy is created, and its responsibility flag for the recipient is set to true; all other responsibility flags are set to false.

b) The necessary trace information (indicating *rerouted* and the name of the MTAE to which the MPDU could not be transferred) is added to all MPDU copies being rerouted.

c) An RT-TRANSFER.request is issued for each UMPDU. Before transferring the MPDU, recipient information whose responsibility flags are false may be removed, unless Disclosure of Other Recipients was requested.

II) If the MPDU is a User or Probe MPDU then, for each recipient for whom no alternate route exists, or for the entire MPDU if the MTAE does not perform alternate routing, a Delivery Report MPDU with the appropriate trace information is generated, and an RT-TRANSFER.request is issued for this Delivery report MPDU. Any copy which cannot be routed is then discarded.

III) If the MPDU is a Delivery Report MPDU, and no alternate route exists, the MPDU is discarded. The transfer may also be logged for accounting and/or diagnostic purposes.

4.1.4 Specification of the MPDU

The unit of exchange between MTAEs is a Message Protocol Data Unit (MPDU), which contains the information required to relay either a message, probe, or delivery report. MPDUs are transferred by means of the RTS, which uses session/presentation services to prevent loss of all or any part of an MPDU. The manner in which MPDUs are transferred by means of the RTS is described in clause 4.1.9 below.

The formal specification of P1 is a module (see CCITT Rec. X.409) that contains the MPDU definitions. The structure of MPDUs is thus defined using the notation specified in CCITT Rec. X.409. The binary representation of each MPDU can be inferred from its notational description with the aid of the referenced document. Data elements identified as OPTIONAL either are "conditional" parameters of the layer service primitives or represent points where flexibility in the internal operation of the MTS is allowed.

As is evident from the following definitions, two classes of MPDU are defined. User MPDUs carry messages toward one or more of their intended recipient UAs. Service MPDUs carry information about messages. Two types of Service MPDU are currently defined. A Delivery Report MPDU carries a delivery or non-delivery report toward the originating UA. A Probe MPDU carries a request for information about the deliverability of a message toward one or more potential recipient UAs.

Every MPDU has two parts, Envelope and Content. The Envelope contains the information the MTS requires to route the MPDU to its intended destination. Information in the envelope may be changed, added to, or deleted during the MPDU's passage through the MTS. The Content is the primary information the MPDU is intended to convey. In general, it is carried unmodified to its destination.

4.1.4.1 Formal Definition of User MPDU

```
P1 DEFINITIONS ::=
BEGIN

MPDU               ::=  CHOICE {[0] IMPLICIT User MPDU, Service MPDU}

ServiceMPDU        ::=  CHOICE {[1] IMPLICIT DeliveryReportMPDU,
                             [2] IMPLICIT ProbeMPDU}

UserMPDU           ::=  SEQUENCE {UMPDUEnvelope, UMPDUContent}

UMPDUEnvelope      ::=  SET {
                             MPDUIdentifier,
                             originator ORName,
                             original EncodedInformationTypes OPTIONAL,
                             ContentType,
                             UAContentID OPTIONAL,
                             Priority DEFAULT normal,
                             PerMessageFlag DEFAULT {},
                             deferredDelivery [0] IMPLICIT Time OPTIONAL,
                             [1] IMPLICIT SEQUENCE OF PerDomainBilateralInfo OPTIONAL,
                             [2] IMPLICIT SEQUENCE OF RecipientInfo,
                             TraceInformation,
                             latestDelivery [3] IMPLICIT Time OPTIONAL,
                             InternalTraceInfo OPTIONAL}

UMPDUContent       ::=  OCTET STRING

-- time

Time               ::=  UTCTime
```

```
-- various envelope information

MPDUIdentifier          ::=  [APPLICATION 4] IMPLICIT SEQUENCE {
                                  GlobalDomainIdentifier, IA5String}

ContentType             ::=  [APPLICATION 6] IMPLICIT INTEGER {p2(2)}

UAContentID             ::=  [APPLICATION 10] IMPLICIT PrintableString

Priority                ::=  [APPLICATION 7] IMPLICIT INTEGER {
                                  normal(0), deferrable(1), urgent(2)}

PerMessageFlag          ::=  [APPLICATION 8] IMPLICIT BIT STRING {
                                  discloseRecipients(0),
                                  conversionProhibited(1),
                                  alternateRecipientAllowed(2),
                                  contentReturnRequest(3)}

-- per-domain bilateral information

PerDomainBilateralInfo  ::=  SEQUENCE {
                                  CountryName,
                                  AdministrationDomainName,
                                  BilateralInfo}

BilateralInfo           ::=  ANY

-- recipient information

RecipientInfo           ::=  SET {
                                  recipient ORName,
                                  [0] IMPLICITExtensionIdentifier,
                                  [1] IMPLICIT PerRecipientFlag,
                                  alternateRecipient [3] IMPLICIT ORName OPTIONAL -- none if
                                  absent --}

ExtensionIdentifier     ::=  INTEGER

PerRecipientFlag        ::=  BIT STRING -- see Figure 17.

-- trace information

TraceInformation        ::=  [APPLICATION 9] IMPLICIT SEQUENCE OF
                                  SEQUENCE {GlobalDomainIdentifier, DomainSuppliedInfo}

DomainSuppliedInfo      ::=  SET {
                                  arrival  [0] IMPLICIT Time,
                                  deferred [1] IMPLICIT Time OPTIONAL,
                                  action [2] IMPLICIT INTEGER {relayed(0), rerouted(1)},
                                  converted EncodedInformationTypes OPTIONAL,
                                  previous GlobalDomainIdentifier OPTIONAL}

InternalTraceInfo       ::=  [ PRIVATE 0 ] IMPLICIT SEQUENCE OF PerMTATraceInfo

PerMTATraceInfo         ::=  SEQUENCE {
                                  MTAName,
                                  MTASuppliedInfo}

MTASuppliedInfo         ::=  SET {
                                  arrival [0] IMPLICIT Time,
                                  deferred [1] IMPLICIT Time OPTIONAL,
                                  action [2] IMPLICIT INTEGER {relayed (0), rerouted (1)}
                                  previous MTAName OPTIONAL}
```

```
-- global domain identifier
GlobalDomainIdentifier      ::=  [APPLICATION 3] IMPLICIT SEQUENCE {
                                       CountryName,
                                       AdministrationDomainName,
                                       PrivateDomainIdentifier OPTIONAL}

CountryName                 ::=  [APPLICATION 1] CHOICE {
                                       NumericString,
                                       PrintableString}

AdministrationDomainName    ::=  [APPLICATION 2] CHOICE {
                                       NumericString,
                                       PrintableString}

PrivateDomainIdentifier     ::=  CHOICE {
                                       NumericString,
                                       PrintableString}

-- O/R name
ORName                      ::=  [APPLICATION 0] IMPLICIT SEQUENCE{
                                       StandardAttributeList,
                                       DomainDefinedAttributeList OPTIONAL}

StandardAttributeList       ::=  SEQUENCE{
                                       CountryName OPTIONAL,
                                       AdministrationDomainName OPTIONAL,
                                       [0] IMPLICIT  X121Address OPTIONAL,
                                       [1] IMPLICIT TerminalID OPTIONAL,
                                       [2] PrivateDomainName OPTIONAL,
                                       [3] IMPLICIT OrganizationName OPTIONAL,
                                       [4] IMPLICIT UniqueUAIdentifier OPTIONAL,
                                       [5] IMPLICIT PersonalName OPTIONAL,
                                       [6] IMPLICIT SEQUENCE OF OrganizationalUnit OPTIONAL,
                                       [7] IMPLICIT PositionorRole  OPTIONAL,
                                       GeographicalAttributes  OPTIONAL}

DomainDefinedAttributeList  ::=  SEQUENCE OF DomainDefinedAttribute

DomainDefinedAttribute      ::=  SEQUENCE{
                                       type PrintableString,      -- type = "MIDA"
                                       value PrintableString}   -- value  =  MTAName:UALocalId

MTAName                     ::=  PrintableString

X121Address                 ::=  NumericString

TerminalID                  ::=  PrintableString

PositionorRole              ::=  PrintableString

GeographicalAttributes      ::=  [ PRIVATE 1]  IMPLICIT SET {
                                       streetNameNumber [0] IMPLICIT PrintableString OPTIONAL,
                                       townName [1] IMPLICIT PrintableString OPTIONAL,
                                       regionName [2] IMPLICIT PrintableString OPTIONAL}

OrganizationName            ::=  PrintableString

UniqueUAIdentifier          ::=  NumericString
```

```
PersonalName              ::=  SET {
                                 surName [0] IMPLICIT PrintableString,
                                 givenName [1] IMPLICIT PrintableString OPTIONAL,
                                 initials [2] IMPLICIT PrintableString OPTIONAL,
                                 generationQualifier [3] IMPLICIT PrintableString OPTIONAL,
                                 title [4] IMPLICIT PrintableString OPTIONAL}
OrganizationalUnit        ::=  PrintableString
PrivateDomainName         ::=  CHOICE {
                                 NumericString,
                                 PrintableString}
-- encoded information types
EncodedInformationTypes   ::=  [APPLICATION 5] IMPLICIT SET {
                                 [0] IMPLICIT BIT STRING {
                                     undefined(0), tLX(1), iA5Text(2), g3Fax (3),
                                     tIF0(4), tTX(5), videotex(6),voice(7), sFD(8),
                                     tIF1(9), oDIF(10), iSO6937(11)}
                                 [1] IMPLICIT G3NonBasicParams OPTIONAL,
                                 [2] IMPLICIT TeletexNonBasicParams OPTIONAL,
                                 [3] IMPLICIT PresentationCapabilities OPTIONAL}
G3NonBasicParams          ::=  BIT STRING {
                                 twoDimensional(8),
                                 fineResolution(9),
                                 unlimitedLength(20),
                                 b4Length(21),
                                 a3Width(22),
                                 b4Width(23),
                                 uncompressed(30)}
TeletexNonBasicParams     ::=  SET {
                                 graphicCharacterSets [0] IMPLICIT T61String OPTIONAL,
                                 controlCharacterSets [1] IMPLICIT T61String OPTIONAL,
                                 pageFormats [2] IMPLICIT OCTET STRING OPTIONAL,
                                 miscTerminalCapabilities [3] IMPLICIT T61String OPTIONAL,
                                 privateUse [4] IMPLICIT OCTET STRING OPTIONAL}
PresentationCapabilities  ::=  T73. PresentationCapabilities
```

4.1.4.2 Formal Definition of Delivery Report MPDU

```
DeliveryReportMPDU          ::=  SEQUENCE {
                                      DeliveryReportEnvelope, DeliveryReportContent}

DeliveryReportEnvelope      ::=  SET {
                                      report MPDUIdentifier,
                                      originator ORName,
                                      TraceInformation,
                                      InternalTraceInfo OPTIONAL}

DeliveryReportContent       ::=  SET {
                                      original MPDUIdentifier,
                                      intermediate TraceInformation OPTIONAL,
                                      Intermediate InternalTraceInfo OPTIONAL,
                                      UAContentId OPTIONAL,
                                      [0] IMPLICIT SEQUENCE OF ReportedRecipientInfo,
                                      returned [1] IMPLICIT UMPDUContent OPTIONAL,
                                      billingInformation [2] ANY OPTIONAL}

ReportedRecipientInfo       ::=  SET {
                                      recipient [0] IMPLICIT ORName,
                                      [1] IMPLICIT ExtensionIdentifier,
                                      [2] IMPLICIT PerRecipientFlag,
                                      [3] IMPLICIT LastTraceInformation,
                                      intendedRecipient [4] IMPLICIT ORName OPTIONAL,
                                      [5] IMPLICIT SupplementaryInformation OPTIONAL}

-- last trace information

LastTraceInformation        ::=  SET {
                                      arrival [0] IMPLICIT Time,
                                      converted EncodedInformationTypes OPTIONAL,
                                      [1] Report}

Report                      ::=  CHOICE {
                                      [0] IMPLICIT DeliveredInfo,
                                      [1] IMPLICIT NonDeliveredInfo}

DeliveredInfo               ::=  SET {
                                      delivery [0] IMPLICIT Time,
                                      typeOfUA [1] IMPLICIT INTEGER {
                                           public(0), private(1)}}

NonDeliveredInfo            ::=  SET {
                                      [0] IMPLICIT ReasonCode,
                                      [1] IMPLICIT DiagnosticCode OPTIONAL}

ReasonCode                  ::=  INTEGER {
                                      transferFailure(0),
                                      unableToTransfer(1),
                                      conversionNotPerformed(2),
                                      latestDeliveryTimePassed (3)}
```

```
DiagnosticCode              ::=  INTEGER {
                                     unrecognizedORName(0),
                                     ambiguousORName(1),
                                     mtaCongestion(2),
                                     loopDetected(3),
                                     uaUnavailable(4),
                                     maximumTimeExpired(5),
                                     encodedInformationTypesUnsupported(6),
                                     contentTooLong(7),
                                     conversionImpractical(8),
                                     conversionProhibited(9),
                                     implicitConversionNotRegistered(10),
                                     invalidParameters(11)}

-- supplementary information

SupplementaryInformation    ::=  PrintableString -- length limited and for further study --
```

4.1.4.3 Formal Definition of Probe MPDU

```
ProbeMPDU                   ::=  ProbeEnvelope

ProbeEnvelope               ::=  SET {
                                     probe MPDUIdentifier,
                                     originator ORName,
                                     ContentType,
                                     UAContentID OPTIONAL,
                                     original EncodedInformationTypes OPTIONAL,
                                     TraceInformation,
                                     PerMessageFlag DEFAULT {},
                                     contentLength [0] IMPLICIT INTEGER OPTIONAL,
                                     [1] IMPLICIT SEQUENCE OF PerDomainBilateralInfo OPTIONAL,
                                     [2] IMPLICIT SEQUENCE OF RecipientInfo,
                                     InternalTraceInfo OPTIONAL}

END -- of P1 DEFINITIONS --
```

4.1.5 Common Data Types

A number of data types figure in the definition of several MPDUs. These data types are defined in the following subsections.

4.1.5.1 O/R Name

An O/R name specifies the originator or recipient of a message according to the principles of naming and addressing laid out in Section 2. An O/R name comprises a number of standard attributes, optionally followed by a number of attributes defined by the Management Domain to which the originator or recipient subscribes:

```
ORName ::= [APPLICATION 0] IMPLICIT SEQUENCE {
    StandardAttributeList,
    DomainDefinedAttributeList OPTIONAL}
```

The standard attributes used in any O/R name are selected from those mentioned in Section 2:

```
StandardAttributeList :: = SEQUENCE {
    CountryName OPTIONAL,
    AdministrationDomainName OPTIONAL,
    [0] IMPLICIT  X121Address OPTIONAL,
    [1] IMPLICIT TerminalID OPTIONAL,
    [2] PrivateDomainName OPTIONAL,
    [3] IMPLICIT OrganisationName OPTIONAL,
    [4] IMPLICIT UniqueUAIdentifier OPTIONAL,
    [5] IMPLICIT PersonalName OPTIONAL,
    [6] IMPLICIT SEQUENCE OF OrganizationalUnit OPTIONAL,
    [7] IMPLICIT PositionorRole  OPTIONAL,
    GeographicalAttributes  OPTIONAL}
```

Although the formal definition above allows any combination of standard and domain-defined attributes in an O/R name, only those combinations listed below are actually allowed in the CCITT Recommendations:

I) Telematic address:

- X121 Address
- TerminalID (optional)

II) Message handling address (variant 1):

- CountryName
- AdministrationDomainName
- UniqueUAIdentifier
- DomainDefinedAttributeList (optional)

III) Message handling address (variant 2):

- CountryName
- AdministrationDomainName
- X121Address
- DomainDefinedAttributeList (optional)

IV) Message handling address (variant 3):

- CountryName
- AdministrationDomainName

One or more of the following attributes:

- PrivateDomainName
- OrganizationName
- PersonalName
- OrganizationalUnit (perhaps several)
- DomainDefinedAttributeList (optional)

An X121Address is constructed in accordance with CCITT Rec. X.121:

```
X121Address :: = NumericString -- see CCITT X.121
```

A TerminalID is a Telex Answer back or a Teletex terminal identifier:

```
TerminalID :: = PrintableString
```

A CountryName is a Numeric or Printable String encoding a country code specified by CCITT Rec. X.121, or by ISO 3166, respectively:

```
CountryName :: = [APPLICATION 1] CHOICE {
   NumericString, -- see CCITT X.121 --
   PrintableString -- see ISO 3166 --}
```

An AdministrationDomainName identifies a public management domain within a country by means of a Numeric or Printable String taken from a defined list administered by that country:

```
AdministrationDomainName :: = [APPLICATION 2] CHOICE {
   NumericString,
   PrintableString}
```

A UniqueUAIdentifier unambiguously identifies a user within the scope of a public management domain:

```
UniqueUAIdentifier :: = NumericString
```

A PrivateDomain Name identifies a private management domain, which may be connected to a public management domain:

```
PrivateDomainName :: = CHOICE {
   NumericString,
   PrintableString}
```

An OrganizationName identifies the user's organization:

```
OrganizationName :: = PrintableString
```

A PersonalName specifies the user's surname, given name (optional), initials (optional), generational qualifier (optional; for example, "Jr."), and title (optional; for example, "Dr."):

```
PersonalName :: = SET {
   surName [0] IMPLICIT PrintableString,
   givenName [1] IMPLICIT PrintableString OPTIONAL,
   initials [2] IMPLICIT PrintableString OPTIONAL,
   generationQualifier [3] IMPLICIT PrintableString OPTIONAL,
   title [4] IMPLICIT PrintableString OPTIONAL}
```

Note 42
The order of the members of this set may have significance to a directory service.

An OrganizationalUnit identifies a unit (for example, a department) of the user's organization:

```
OrganizationalUnit :: = PrintableString
```

Note 43
The order of members of this sequence may have significance to a directory service.

A PositionorRole identifies a user's function in an organization:

```
PositionorRole :: = PrintableString
```

The GeographicalAttributes identify the user by reference to his geographical location:

```
GeographicalAttributes :: = [PRIVATE 1] IMPLICIT SET {
   streetNameNumber [0] IMPLICIT PrintableString OPTIONAL,
    townName [1] IMPLICIT PrintableString OPTIONAL,
    regionName [2] IMPLICIT PrintableString OPTIONAL}
```

A DomainDefinedAttributeList provides the type and value of non-standard attributes defined by a user's management domain:

```
DomainDefinedAttributeList :: = SEQUENCE OF DomainDefinedAttribute

DomainDefinedAttribute :: = SEQUENCE {
    type PrintableString,
    value PrintableString}
```

MIDA uses the Message Handling Address Variant 3, as below:

```
ORName:: = [APPLICATION 0] IMPLICIT SEQUENCE{

    SEQUENCE{
        CountryName,
        AdministrationDomianName,
        [2]  PrivateDomainName,
        [3]  IMPLICIT OrganizationName OPTIONAL,
        [5]  IMPLICIT PersonalName OPTIONAL,
        [6]  IMPLICIT SEQUENCE OF OrganizationalUnit OPTIONAL}

    SEQUENCE OF
        SEQUENCE{
        type PrintableString  =  "MIDA"
        value PrintableString  =  MTAName:UALocalId}}
```

4.1.5.2 Extension Identifier

An extension identifier provides a shorthand method for identifying an individual recipient within the context of a particular message. When combined with an MPDU identifier, it also unambiguously identifies a copy of the original User or Probe MPDU in a way that is more convenient and reliable than is possible using O/R names. Values are assigned from the range that begins with one and ends with the number of recipients of the message as submitted:

```
ExtensionIdentifier :: = INTEGER
```

4.1.5.3 Global Domain Identifier

A Global Domain Identifier unambiguously identifies a management domain within the scope of the entire message handling system. It is used to identify the source of trace information, Delivery Report MPDUs, etc. In the case of a public management domain, it comprises the domain's country and public domain names. In the case of a private management domain, it comprises the country and domain names of the associated public domain, plus a private domain identifier. The Private Domain Identifier unambiguously identifies the private domain within the scope of the associated public domain; it may be identical to the private domain's PrivateDomainName:

```
GlobalDomainIdentifier :: = [APPLICATION 3] IMPLICIT SEQUENCE {
    CountryName,
    AdministrationDomainName,
    PrivateDomainIdentifier OPTIONAL}
PrivateDomainIdentifier :: = CHOICE {
   NumericString,
   PrintableString}
```

4.1.5.4 Encoded Information Types

The Encoded Information Types of a message are the kind(s) of information that appear in its content. Both the basic encoded information types present, and any non-basic parameters that are required, are specified:

```
EncodedInformationTypes :: = [APPLICATION 5] IMPLICIT SET {
   [0] IMPLICIT BIT STRING {
       undefined(0), tLX(1), iA5text(2), g3Fax(3), tIF0(4),
       tTX(5), videotex(6), voice(7), sFD(8), tIF1(9), oDIF(10), iSO6937text(11)}
   [1] IMPLICIT G3NonBasicParams OPTIONAL,
   [2] IMPLICIT TeletexNonBasicParams OPTIONAL,
   [3] IMPLICIT PresentationCapabilities OPTIONAL}
```

The undefined type is any type not specified by this Standard. The oDIF type is specified in ECMA-xx. The iSO6937text type is described in clause 4.2.3.2 . The remaining types are described in CCITT Rec. X.411.

4.1.5.5 Trace Information

Trace information documents the passage of an MPDU between domains. It describes the action(s) taken by each management domain through which the MPDU passes. Each MD adds its Trace Information at the end of the existing sequence of Trace Information.

Each element of trace information includes the GlobalDomainIdentifier (see clause 4.1.5.3) of the management domain supplying the information:

```
TraceInformation :: = [APPLICATION 9] IMPLICIT SEQUENCE OF
   SEQUENCE {
       GlobalDomainIdentifier,
       DomainSuppliedInfo}
```

Each element of trace information includes the date and time the MPDU arrived in the management domain. In the originating domain's case, this is the submission time. Additionally, if the Deferred Delivery service caused the domain to hold the message for a period of time, the time when the domain started to process the message for delivery or fowarding is also included. The trace information also specifies the action the domain took with respect to the MPDU. Relayed is the normal action of a relay MTAE. Rerouted indicates that an attempt had previously been made to route the MPDU to the domain identified by the previous GlobalDomainIdentifier. If the domain subjects a User MPDU to conversion, the result of the conversion is also included:

```
DomainSuppliedInfo :: = SET {
   arrival [0] IMPLICIT Time,
   deferred [1] IMPLICIT Time OPTIONAL,
   action [2] IMPLICIT INTEGER {relayed(0), rerouted(1)},
   converted EncodedInformationTypes OPTIONAL,
   previous GlobalDomainIdentifier OPTIONAL}
```

In connection with the converted EncodedInformationTypes parameter, it should be noted that MIDA does not support encoded information type conversion of any kind.

4.1.5.6 MPDU Identifier

An MPDU identifier is used to distinguish one MPDU from another within the MTS. The originating MTAE assigns a string identifier to each MPDU it creates, choosing the identifier in such a way that it unambiguously identifies, within the originating management domain, the particular message submission event, probe event, or delivery report generation that gave rise to the MPDU. When combined with the originating domain's global domain identifier, the string identifier forms the MPDU identifier, which unambiguously identifies the submission event, probe event, or delivery report generation throughout the entire MTS:

```
MPDUIdentifier :: = [APPLICATION 4] IMPLICIT SEQUENCE {
   GlobalDomainIdentifier,
   IA5String}       --  (It is the responsibility of each MTA to generate unique MPDU
                        Identifiers for the messages submitted to the MTA.)
```

When a User or Probe MPDU is copied for routing to recipients reachable by different paths through the MTS, each copy bears the MPDU identifier of the original MPDU. The copies can be distinguished from one another by means of the responsibility flags and the corresponding extension identifiers, which specify to which recipient(s) each copy is to be delivered.

4.1.5.7 Content Type

A Content Type is supplied by the originating UA and identifies the convention that governs the structure of the message's content. The only defined value identifies the P2 protocol specified in clause 4.2:

```
ContentType :: = [APPLICATION 6] IMPLICIT INTEGER {p2(2)}
```

4.1.5.8 UA Content Identifier

A UA Content Identifier may be provided by the UA and carried back to the originator (in a Delivery Report MPDU) by the MTS. The length of this data element is limited to 16 characters in CCITT Recommendations.

```
UAContentID :: = [APPLICATION 10] IMPLICIT PrintableString
```

4.1.5.9 Time

A Time identifies a calendar date and time of day to be conveyed by means of the protocol. It must include the time of day in terms of UTC (Universal Coordinated Time), and may optionally convey the local time when it is generated. The precision of the Time is to either one second or one minute, as selected by the entity that generates it.

```
Time :: = UTCTime
```

4.1.5.10 Internal Trace Information

InternalTraceInfo documents the passage of an MPDU within a management domain through the MTS. It describes the action taken by each MTA through which it passes. Each MTA adds its trace information at the end of the existing sequence of InternalTraceInfo. Each element of InternalTraceInfo includes the MTAName of the MTA supplying the information. The elements of InternalTraceInfo may be removed before an MPDU leaves a management domain.

```
InternalTraceInfo :: = [ PRIVATE 0 ] IMPLICIT SEQUENCE OF PerMTATraceInfo

PerMTATraceInfo :: = SEQUENCE {
                      MTAName,
                      MTASuppliedInfo}

MTAName :: = PrintableString               -- Unique within a Management Domain

MTASuppliedInfo :: = SET {
                      arrival [0] IMPLICIT Time,
                      deferred [1] IMPLICIT Time OPTIONAL,
                      action [2] IMPLICIT INTEGER {relayed (0), rerouted (1)}
                      previous MTAName OPTIONAL}
```

4.1.6 User MPDU

A User MPDU carries a message toward one or more of its intended recipient UAs. It has two parts, envelope and content:

UserMPDU :: = SEQUENCE {UMPDUEnvelope, UMPDUContent}

The UMPDU envelope contains the information the MTS requires to route the message to its intended recipients. The UMPDU content is the UA-supplied information the message is intended to convey.

4.1.6.1 UMPDU Envelope

The UMPDU Envelope contains a number of parameters. It is constructed when the message is submitted, and may be changed as the User MPDU makes its way through the MTSL:

```
UMPDUEnvelope :: = SET {
    MPDUIdentifier,
    originator ORName,
    original EncodedInformationTypes OPTIONAL,
    ContentType,
    UAContentID OPTIONAL,
    Priority DEFAULT normal,
    PerMessageFlag DEFAULT {},
    deferredDelivery [0] IMPLICIT Time OPTIONAL,
    [1] IMPLICIT SEQUENCE OF PerDomainBilateralInfo OPTIONAL,
    [2] IMPLICIT SEQUENCE OF RecipientInfo,
    TraceInformation,
    latestDelivery [3] IMPLICIT Time OPTIONAL,
    InternalTraceInfo  OPTIONAL}
```

The MPDUIdentifier parameter distinguishes the message submission event that produced this MPDU from all other message submission events, probe events, or delivery report generations (see clause 4.1.5.6).

The originator parameter identifies the originator of the message carried by the MPDU (see clause 4.1.5.1).

The original EncodedInformationTypes parameter is the value of the MT-SUBMIT.request's encoded-info.-types parameter. The absence of this parameter indicates that the encoded information types are undefined (see clause 4.1.5.4).

The ContentType parameter identifies the type of the message's content (see clause 4.1.5.7).

The UAContentID parameter allows the originator to supply an identifier for the content. This is not delivered to the recipient, but is returned to the originator with any notify event.

The Priority parameter specifies the relative priority of the message carried by the MPDU:

Priority :: = [APPLICATION 7] IMPLICIT INTEGER {normal(0), deferrable(1), urgent(2)}

The PerMessageFlag parameter, a Bit String, specifies options that apply to all recipients of the message. The discloseRecipients bit indicates whether the O/R names of all recipients should be indicated to each recipient UA when the message carried by the MPDU is delivered. The conversionProhibited bit indicates whether conversion prohibition was requested in the MT-SUBMIT.Request primitive. If this bit is not set, implicit conversion may be carried out. The alternateRecipientAllowed bit indicates whether the Alternate Recipient Allowed service was requested in the MT-SUBMIT.Request primitive. The contentReturnRequest bit indicates whether the content of the message is to be returned with any non-delivery notification:

```
PerMessageFlag :: = [APPLICATION 8] IMPLICIT BIT STRING {
    discloseRecipients(0),
    conversionProhibited(1),
    alternateRecipientAllowed(2),
    contentReturnRequest(3)}
```

<u>*Note 44*</u>
MIDA does not support conversion.

The deferredDeliveryTime parameter allows the Deferred Delivery service to be supported by an MTAE other than the originating one, provided that there is an agreement to do so.

The PerDomainBilateralInfo parameter is information supplied by the originating management domain and intended for one of the domains the message will reach as it passes through the MTSL. More than one element of per-domain bilateral information may be present, and each is addressed to a particular domain:

```
PerDomainBilateralInfo :: = SEQUENCE {
    CountryName,
    AdministrationDomainName,
    BilateralInfo}
```

<u>*Note 45*</u>
This element is retained for compatibility with CCITT.

```
BilateralInfo :: = ANY
```

The RecipientInfo parameter contains those pieces of envelope information that may vary from one recipient to another. The first two pieces of such information are the O/R name and extension identifier of the recipient to whom the rest of the information applies:

```
RecipientInfo :: = SET {
    recipient ORName,
    [0] IMPLICIT ExtensionIdentifier,
    [1] IMPLICIT PerRecipientFlag,
    alternateRecipient [3] IMPLICIT ORName OPTIONAL -- none if absent --}

        PerRecipientFlag :: = BIT STRING -- see Figure 17.--
```

Responsibility Flag (Bit 0)	
Off:	0
On:	1

Report Request (Bits 1-2)	
Not used:	00
Basic:	01
Confirmed:	10
Audit-and-confirmed:	11

User Report Request (Bits 3-4)	
No report:	00
Basic:	01
Confirmed:	10
Not used:	11

Deliver To Request (Bits 6-7)	
Intended Recipient:	00
Orig. Spec. Alt. Rec.:	01
Domain Def. Alt. Rec.:	10
Not used:	11

Bits 5 to 7 must be set to ZERO between domains.

Figure 17. Structure of Per-recipient Flag

The bits of the PerRecipientFlag are assigned as shown in Figure 17. The Report Request bits, set by the originating MTAE, request one of the following three kinds of reports:

<u>Basic:</u>	A delivery report is returned only in case of non-delivery, and it contains only the last item of trace information.
<u>Confirmed:</u>	A delivery report is returned in every case, and it contains only the last item of trace information.
<u>Audit-and-confirmed:</u>	A delivery report is returned in every case, and it contains all of the trace information.

The User Report Request bits indicate the kind of report, if any, the originator selected; the binary value 00 indicates that he requested the Prevention of Non-delivery Notification service. The Report Request bits must specify at least the report level requested by the originator. If the Responsibility flag is set, the receiving MTAE shall have the responsibility to either deliver the message carried by the MPDU to this particular recipient, or to relay it to another MTAE for subsequent delivery.

The Deliver To Request bits, when the Responsibility Flag is on, indicate to an MTAE where this message should be delivered, choices are:

<u>Intended Recipient:</u>

To the originally specified recipient

<u>Originator-specified Alternate Recipient:</u>

To the originator requested alternate recipient. The request is set as a result of failure to deliver to the original recipient

<u>Domain-defined Alternate Recipient:</u>

To the (domain defined) alternate recipient.This request may be set as a result of failure to deliver to the original recipient, or to a designated alternate recipient if the domain has defined a "dead letter" recipient.

The presence of the alternateRecipient parameter indicates that the message is to be delivered to the indicated user if it cannot be delivered to the intended recipient. When RecipientInfo appears in a ProbeEnvelope, this parameter is omitted.

The TraceInformation parameter records information about the MPDU's passage between domains (see clause 4.1.5.5).

The InternalTraceInfo parameter records information about the MPDU's passage within a domain (see clause 4.1.5.10).

The latestDelivery parameter specifies the latest time at which the message may be delivered. If this time constraint cannot be achieved, a non-delivery report is returned to the originator.

4.1.6.2 <u>UMPDU Content</u>

The UMPDU content is the UA-supplied information the message is intended to convey. Within a MIDA System, the UMPDU content is not modified by the MTSL but rather is passed transparently through it:

```
UMPDUContent :: = OCTET STRING
```

<u>Note 46</u>
Although the UMPDU Content is passed transparently, the presentation of the UMPDU Content is not necessarily identical on each relaying; e.g. different presentation entities relaying the UMPDU Content may give different length to the primitive elements of the OCTET STRING, if the UMPDU Content is encoded as a constructor.

4.1.7 Delivery Report MPDU

A Delivery Report MPDU carries a delivery or non-delivery report toward the originating UA. It has two parts, envelope and content:

```
DeliveryReportMPDU :: = SEQUENCE {
    DeliveryReportEnvelope, DeliveryReportContent}
```

The delivery report envelope contains the information the MTSL requires to route the report to the originator. The delivery report content is the system-supplied information the report is intended to convey. It may optionally also contain a parameter that is used to return billing information to the originating management domain. An indication of whether the message was delivered to publically or privately owned equipment is also returned.

4.1.7.1 Delivery Report Envelope

The Delivery Report Envelope contains a number of parameters. It is constructed when the report is originated, and may be changed as the Delivery Report MPDU makes its way through the MTSL:

```
DeliveryReportEnvelope :: = SET {
  report MPDUIdentifier,
  originator ORName,
  TraceInformation,
  InternalTraceInfo OPTIONAL}
```

The report MPDUIdentifier parameter distinguishes this MPDU (not the subject User MPDU, the fate of which is being reported) from all other MPDUs (see clause 4.1.5.6).

The originator parameter identifies the originator of the subject message (see clause 4.1.5.1).

The TraceInformation parameter records information about the passage of this MPDU (not the subject User MPDU) between domains (see clause 4.1.5.5).

The InternalTraceInfo parameter records information about the passage of this MPDU (not the subject User MPDU) within a domain (see clause 4.1.5.10).

4.1.7.2 Delivery Report Content

The Delivery Report Content is the system-supplied information the report is intended to convey. It includes the MPDUIdentifier of the subject User MPDU. If an audit-and-confirmed delivery report was requested, it also includes the trace information present in the subject User MPDU when it reached the management domain generating the report. If a content-returned non-delivery report was requested, the content of the report also includes the content of the subject User MPDU:

```
DeliveryReportContent :: = SET {
  original MPDUIdentifier,
  intermediate TraceInformation OPTIONAL,
  UAContentId OPTIONAL,
  [0] IMPLICIT SEQUENCE OF ReportedRecipientInfo,
  returned [1] IMPLICIT UMPDUContent OPTIONAL,
  billingInformation [2] ANY OPTIONAL,
  intermediate InternalTraceInfo OPTIONAL}
```

The ReportedRecipientInfo parameter contains those pieces of information that may vary from one recipient to another. Normally the first two pieces of such information are the O/R name and extension identifier of the recipient to whom the rest of the information applies. In case of an alternate recipient the ExtensionIdentifier relates to the intended recipient.

```
ReportedRecipientInfo :: = SET {
   recipient [0] IMPLICIT ORName,
   [1] IMPLICIT ExtensionIdentifier,
   [2] IMPLICIT PerRecipientFlag,
   [3] IMPLICIT LastTraceInformation,
   intendedRecipient [4] IMPLICIT ORName OPTIONAL,
   [5] IMPLICIT SupplementaryInformation OPTIONAL}
```

The PerRecipientFlag parameter specifies the report requested by the originator (see clause 4.1.5.1).

The LastTraceInformation parameter indicates the fate of the subject message. The information provided includes the date and time at which the subject User MPDU entered the management domain or MTA making the report and, if the message underwent conversion, the result of that conversion:

```
LastTraceInformation :: = SET {
   arrival [0] IMPLICIT Time,
   converted EncodedInformationTypes OPTIONAL,
   [1] Report}
```

If delivery succeeded, the date and time at which the message was delivered to the recipient UA, as well as an indication of whether the UA is publically or privately owned, are specified. If delivery failed, the reason and some diagnostic information are given:

```
Report :: = CHOICE {
   [0] IMPLICIT DeliveredInfo,
   [1] IMPLICIT NonDeliveredInfo}

DeliveredInfo :: = SET {
   delivery [0] IMPLICIT Time,
   typeOfUA [1] IMPLICIT INTEGER {public(0), private(1)}}

NonDeliveredInfo :: = SET {
   [0] IMPLICIT ReasonCode,
   [1] IMPLICIT DiagnosticCode OPTIONAL}
```

The ReasonCode indicates the reason for non-delivery. TransferFailure indicates that a communication failure prevented the MTAE from relaying or delivering the message. UnableToTransfer indicates that the MTAE could not relay or deliver the message because of some problem with the message itself. ConversionNotPerformed indicates that a conversion necessary for the delivery of the message could not be performed. LatestDeliveryTimePassed indicates that the message could not be delivered before the specified latest delivery time:

```
ReasonCode :: = INTEGER {
   transferFailure(0),
   unableToTransfer(1),
   conversionNotPerformed(2),
   latestDeliveryTimePassed(3)}
```

The DiagnosticCode provides further information about the problem:

```
DiagnosticCode :: = INTEGER {
   unrecognizedORName(0),
   ambiguousORName(1),
   mtaCongestion(2),
   loopDetected(3),
   uaUnavailable(4),
   maximumTimeExpired(5),
   encodedInformationTypesUnsupported(6),
   contentTooLong(7),
   conversionImpractical(8),
        conversionProhibited(9),
        implicitConversionNotRegistered(10),
        invalidParameters(11)}
```

The intendedRecipient parameter is present if the subject message was diverted to an alternate recipient. The recipient and intendedRecipient parameters are used in connection with both the Originator-requested Alternate Recipient and Alternate Recipient Allowed services.

The SupplementaryInformation parameter is included for possible future use by PTT supplied gateway mechanisms.

```
SupplementaryInformation :: = PrintableString
```

The contents of the billingInformation parameter is specified by bilateral agreement.

4.1.8 Probe MPDU

A Probe MPDU carries a request for information about the deliverability of a message toward one or more potential recipient UAs. It has just one part, the envelope:

```
ProbeMPDU :: = ProbeEnvelope
```

The Probe Envelope contains the information the MTS requires to route the probe to its destination.

An MTAE may generate a Probe MPDU either to establish that a particular, subject message can be delivered before sending it, or to test in a general way the availability and capabilities of other management domains.

4.1.8.1 Probe Envelope

The Probe Envelope contains a number of parameters. It is constructed when the probe is issued, and may be changed as the Probe MPDU makes its way through the MTS:

```
ProbeEnvelope:: = SET {
    probe MPDUIdentifier,
    originator ORNAME,
    ContentType,
    UAContentID OPTIONAL,
    original EncodedInformationTypes OPTIONAL
    TraceInformation,
    PerMessageFlag DEFAULT{},
    contentLength [0] IMPLICIT INTEGER OPTIONAL,
    [1] IMPLICIT SEQUENCE OF PerDomainBilateralInfo OPTIONAL,
    [2] IMPLICIT SEQUENCE OF RecipientInfo,
    InternalTraceInfo OPTIONAL}
```

The probe MPDUIdentifier parameter distinguishes the Probe event that resulted in the creation of this MPDU from all other probe events, submission events, and delivery report generations (see clause 4.1.5.6).

The originator parameter identifies the originator of the subject message (see clause 4.1.5.1).

The original EncodedInformationTypes parameter is the encoded information types of the subject message. The absence of this parameter indicates that those types are undefined (see clause 4.1.5.4).

The TraceInformation parameter records information about this MPDU's passage between domains (see clause 4.1.5.5).

The InternalTraceInfo parameter records information about this MPDU's passage within a domain (see clause 4.1.5.10).

The PerMessageFlag parameter is as specified for the User MPDU (see clause 4.1.6). However, only the conversionProhibited and alternateRecipientAllowed flags affect the deliverability of the message as tested by Probe.

The ContentLength parameter is the length, in octets, of the content of the subject message.

The PerDomainBilateralInfo parameter is as specified for the User MPDU (see clause 4.1.6). It is included so that the effects of future, bilaterally agreed services, such as Closed User Group and Security Classification, can be checked before actual submission.

The RecipientInfo parameter is identical to that appearing in the UMPDU envelope (see clause 4.1.6). It contains those pieces of envelope information that may vary from one recipient to another. The first two pieces of such information are the O/R name and extension identifier of the recipient to whom the rest of the information applies. Also specified are whether the current MTAE has responsibility for probing delivery to this particular recipient, and the kind of delivery report (if any) requested by the user and by the originating management domain.

4.1.9 Management of Associations

4.1.9.1 Establishing and Releasing Associations

Associations between two MTAEs are created in order that messages may be transferred between them.

Certain features of the associations have to be agreed between the MTAEs; some of these may be negotiated when opening an association, others may be the subject of bilateral agreement. The following features require agreement:

- The maximum number of associations that may exist simultaneously.

  This can be negotiated when opening an association . If the responder MTAE is not prepared to accept another association with the initiator, the connection request is refused.

- Monologue or Two way alternate associations to be created.

  This can be negotiated when opening an association. If the initiator requests two way alternate, and the responder can only support monologue, the RT-OPEN.request is refused.

- Which MTAE has reponsibility for establishing associations .

  This is negotiable when Two way Alternate dialogue is selected. If the responder MTAE wishes to be the one responsible for establishing associations, the RT-OPEN.request is refused.

- Whether associations are permanently established, or established and released when required.

  This feature of the management of the association is resolved on a unilateral or bilateral basis, by means not specified in this version of the MIDA protocol. Note that the actual termination of the association can only be initiated via an RT-CLOSE.request, which cannot be refused by the partner. Unilateral release may take place, for example, when the interconnect mechanism is a dial up circuit.

- Selections of lower level mapping.

  This Standard defines two alternative lower level mappings. Between domains, the Basic Activity Subset Session service must be supported by publicly visible MTAs. Interworking between domains using the Presentation mapping may be used by bilateral agreement. Within a domain, it is the domains management's perogative to define which mapping is to be used between the equipments it controls.

- P1 Protocol selection.

  This Standard defines extensions to the CCITT P1 definitions. A different protocol identifier is therefore required at association establishment time. If the remote domain cannot support the fundamental and basic MIDA P1 services , the connection is refused.

The following default settings are intended to assure a successful association between domains when no bilateral agreements are in force:

I) Maximum number of associations:

1 in each direction.

II) Monologue or Two Way Alternate:

Monologue.

III) MTAE responsible for establishing the association:

MTAE with a message to send.

IV) Permanent on Temporary association:

Temporary.

V) Lower layer mapping:

To Basic Activity Subset Session services.
Transport class 0.

VI) P1 Protocol:

CCITT-compatible.

VII) Naming and Addressing, UAS and MTS:

Any of CCITT-defined variants.

Note 47
Delivery, Notification etc. from the remote domain should contain "correct" naming convention for that domain to be used in future associations. Also, when creating a reply to a remote domain, Naming and Addressing should be as defined in the previously received message.

VIII) RT-OPEN User-data:

Password omitted.

Associations are established by invoking the RT-OPEN.request primitive with the following parameters:

Parameter Name	Parameter Value
Responder-address	The address of the RTS of the corresponding MTAE
Dialogue-mode	Monologue or two-way alternate, according to bilateral agreement, the initiator's preferred choice or the constraints of lower level mapping
Initial-turn	With the initiator if and only if it has MPDUs to send (this only applies to two-way alternate associations)
Application-protocol	P1 or MIDA P1

User-data	**CHOICE {**
	NULL,-- *if no validation is required*
	[1] IMPLICIT SET {
	calling mTAName [0] IMPLICIT IA5String,
	password [1] ANY}}

The mTAName and password are omitted if some other method of validation is used (for example, calling party network address or a permanent circuit). Otherwise, the validation parameters are checked and an RT-OPEN.response issued with the following parameters:

Parameter Name	Parameter Value
Disposition	Accepted or refused
User-data	If the association was accepted, the same as above for the called MTA
Refusal-reason	Unacceptable dialogue mode, Busy, or Validation failure

Associations are released by invoking the RT-CLOSE.request primitive, which has no parameters.

4.1.9.2 Transferring MPDUs

Each MPDU is transferred by invoking the RT-TRANSFER.request primitive with the following parameters:

Parameter Name	Parameter Value
RSDU	The MPDU to be transferred
Transfer-time	Determined by local rule

The following are the rules for assigning MPDUs to associations:

I) Each MPDU is assigned a priority based upon its type and User UMPDU priority in accordance with Table 17.

II) MPDUs are assigned to associations in accordance with their priorities.

III) Several associations may be used to carry MPDUs of the same priority.

IV) On any one association, higher priority MPDUs are sent first.

V) On any one association, messages of the same priority are sent first-in-first-out (FIFO).

Priority	MPDU Type	UMPDU Priority
1	SMPDU or UMPDU	Urgent
2	UMPDU	Normal
3	UMPDU	Deferrable

Table 17. MPDU Priorities

4.1.9.3 Managing the Turn

If the association is two-way alternate, the MTAE without the turn may issue an RT-TURN-PLEASE.request primitive the priority parameter of which reflects the highest priority MPDU awaiting transfer. When the MTAE requests the turn so that it can issue an RT-CLOSE.request primitive, the priority parameter should reflect the lowest-priority MPDU the MTAE is prepared to receive before releasing the association.

The MTAE that has the turn will issue a RT-TURN-GIVE.request primitive in response to an RT-TURN-PLEASE.indication when it has no MPDUs to transfer the priorities of which are equal to or higher than, that indicated in the RT-TURN-PLEASE.indication.

If associations are monologue and MPDU's have to flow in both directions between a pair of RTS-users, different associations are needed.

4.1.9.4 Lower Level Addressing

It is necessary that a mapping be performed from the Application level address attributes to the Transport Address. This transport address may be used to derive a 16-digit TSAP Identifier, and the network address. The transport address is used in the following positions:

- as Called Address and Calling Address in the T-Connect primitives,
- as Called SSAP Address and Calling SSAP Address in the S-Connect primitives,
- as User Called and User Caller Address in the P-Connect primitives (if applicable),
- as Responder Address and Initiator Address in the RT-OPEN primitives,
- as the initiating RTS address in the (session) connection identifier.

The location of the mapping funtion is a local matter within each end system. The externally visable address will comprise the 16-digit TSAP identifier and the global network address.

Note 47
CCITT defines the TSAP identifier to a maximum of 16 digits.

4.2 User Agent Sublayer Protocol (P2)

4.2.1 Introduction

This clause describes the User Agent Protocol (P2), the peer protocol that governs communication between two UAEs in order to provide a user with the UAS services defined in clause 3.2.

The P2 protocol makes use of the MTS services described in clause 3.1. The unit of exchange between two UAEs is a User Agent Protocol Data Unit (UAPDU). UAPDUs are transferred, by means of the MTS, as the contents of messages.

The formal specification of P2 is a module (see CCITT Rec. X.420) that contains the definition of the UAPDU.

As is evident from the following definitions, two types of UAPDU are defined. The User Message UAPDU (UM-UAPDU) represents a User Message. The Status Report UAPDU (SR-UAPDU) represents a notification of the receipt or non-receipt of a User Message. These two types are described below, along with the data types common to the definitions of both.

4.2.1.1 Formal Definition of UM-UAPDU

```
P2 DEFINITIONS ::=
BEGIN

UAPDU                        ::=  CHOICE {
                                        [0] IMPLICIT UM-UAPDU,
                                        [1] IMPLICIT SR-UAPDU}

-- User Message UAPDU

UM-UAPDU                     ::=  SEQUENCE {Heading, Body}

-- heading

Heading                      ::=  SET {
    UserMessageId,
    originator             [0]  IMPLICIT ORDescriptor OPTIONAL,
    authorizingUsers       [1]  IMPLICIT SEQUENCE OF ORDescriptor OPTIONAL,
       -- only if not the originator
    primaryRecipients      [2]  IMPLICIT SEQUENCE OF Recipient OPTIONAL,
    copyRecipients         [3]  IMPLICIT SEQUENCE OF Recipient OPTIONAL,
    blindCopyRecipients    [4]  IMPLICIT SEQUENCE OF Recipient OPTIONAL,
    inReplyTo              [5]  IMPLICIT UserMessageId OPTIONAL,
       -- omitted if not in reply to a previous message
    obsoletes              [6]  IMPLICIT SEQUENCE OF UserMessageId OPTIONAL,
    crossReferences        [7]  IMPLICIT SEQUENCE OF UserMessageId OPTIONAL,
    subject                [8]  CHOICE {T61String} OPTIONAL,
    expiryDate             [9]  IMPLICIT Time OPTIONAL, -- if omitted, expiry date is never
    replyBy               [10]  IMPLICIT Time OPTIONAL,
    replyToUsers          [11]  IMPLICIT SEQUENCE OF ORDescriptor OPTIONAL,
       -- each O/R descriptor must contain an O/R name
    importance            [12]  IMPLICIT INTEGER {low(0), normal(1), high(2)} DEFAULT normal,
    sensitivity           [13]  IMPLICIT INTEGER {personal(1), private(2), companyConfidential(3)}
                                     OPTIONAL,
    autoforwarded         [14]  IMPLICIT BOOLEAN DEFAULT FALSE,
       -- indicates that the forwarded message body part(s) were autoforwarded --
    preparationDate       [15]  IMPLICIT Time OPTIONAL,
    warningDate           [16]  IMPLICIT Time OPTIONAL}
```

```
UserMessageId          ::=  [APPLICATION 11] IMPLICIT SET {
                                  ORName OPTIONAL,
                                  PrintableString}

ORName                 ::=  P1.ORName

ORDescriptor           ::=  SET {-- at least one of the first two members must be present
                                ORName OPTIONAL,
                                freeformName [0] IMPLICIT T61String OPTIONAL,
                                telephoneNumber [1] IMPLICIT PrintableString OPTIONAL}

Recipient              ::=  SET {
                                  [0] IMPLICIT ORDescriptor,
                                  reportRequest [1] IMPLICIT BIT STRING {
                                      receiptNotification(0),
                                      nonReceiptNotification(1),
                                      returnUserMessage(2)} DEFAULT {},
                                      -- if requested, the O/R descriptor must contain
                                      --    an O/R name --
                                  replyRequest [2] IMPLICIT BOOLEAN DEFAULT FALSE
                                      -- if true, the O/R descriptor must contain
                                      --    an O/R name --}

-- body

Body                   ::=  SEQUENCE OF BodyPart

BodyPart               ::=  CHOICE {
                                   [0]  IMPLICIT IA5Text,
                                   [1]  IMPLICIT TLX,
                                   [2]  IMPLICIT Voice,
                                   [3]  IMPLICIT G3Fax,
                                   [4]  IMPLICIT TIF0,
                                   [5]  IMPLICIT TTX,
                                   [6]  IMPLICIT Videotex,
                                   [7]  NationallyDefined,
                                   [8]  IMPLICIT Encrypted,
                                   [9]  IMPLICIT ForwardedUserMessage,
                                  [10]  IMPLICIT SFD,
                                  [11]  IMPLICIT TIF1,
                                  [12]  IMPLICIT ODIF,
                                  [13]  IMPLICIT Transparent,
                                  [14]  IMPLICIT ISO6937Text}

-- body part types

IA5Text                ::=  SEQUENCE {
                                  SET {repertoire [0] IMPLICIT INTEGER {ia5(5), ita2(2)}
                                      DEFAULT ia5
                                      -- additional members of this Set
                                      -- are a possible future extension --},
                                  IA5String}

TLX                    ::=  for further study

Voice                  ::=  SEQUENCE {
                                  SET, -- members are for further study
                                  BIT STRING}

G3Fax                  ::=  SEQUENCE {
                                  SET {
                                      numberOfPages [0] IMPLICIT INTEGER OPTIONAL,
                                      [1] IMPLICIT P1.G3NonBasicParams OPTIONAL},
                                  SEQUENCE OF BIT STRING}
```

```
TIF0                        ::=  T73Document

T73Document                 ::=  SEQUENCE OF T73.ProtocolElement

TTX                         ::=  SEQUENCE {
                                     SET {
                                         numberOfPages [0] IMPLICIT INTEGER OPTIONAL,
                                         telexCompatible [1] IMPLICIT BOOLEAN DEFAULT FALSE,
                                         [2] IMPLICIT P1.TeletexNonBasicParams OPTIONAL},
                                     SEQUENCE OF T61String}

Videotex                    ::=  SEQUENCE {
                                     SET, -- members are for further study
                                     VideotexString}

NationallyDefined           ::=  ANY

Encrypted                   ::=  SEQUENCE {
                                     SET, -- members are for further study
                                     BIT STRING}

ForwardedUserMessage        ::=  SEQUENCE {
                                     SET {
                                         delivery [0] IMPLICIT Time OPTIONAL,
                                         [1] IMPLICIT DeliveryInformation OPTIONAL},
                                     UM-UAPDU}

DeliveryInformation         ::=  SET {
                                       [0] IMPLICIT P1.ContentType,
                                       originator P1.ORName,
                                       original [1] IMPLICIT P1.EncodedInformationTypes,
                                       P1.Priority DEFAULT normal,
                                       [2]  IMPLICIT DeliveryFlags,
                                       otherRecipients [3] IMPLICIT SEQUENCE OF P1.ORName
                                       OPTIONAL,
                                       thisRecipient [4] IMPLICIT P1.ORName,
                                       intendedRecipient [5] IMPLICIT P1.ORName OPTIONAL,
                                       converted [6] IMPLICIT P1.EncodedInformationTypes
                                       OPTIONAL,
                                       submission [7] IMPLICIT P1.Time}

SFD                         ::=  SFD.Document

-- note that SFD and T73Document use the same space of application-wide tags
-- which is different from that used for other MHS protocols

TIF1                        ::=  T73Document

ODIF                        ::=  SEQUENCE OF ODIFProtocolElement -- see ECMA-xx

Transparent                 ::=  ANY

ISO6937Text                 ::=  SEQUENCE {
                                   SET { repetoire[0] IMPLICIT INTEGER {part1and2 (0)} DEFAULT
                                   part1and2}
                                   SEQUENCE OF ISO6937Line}

ISO6937Line                 ::=  [0] IMPLICIT OCTET STRING
```

4.2.1.2 Formal Definition of SR-UAPDU

```
SR-UAPDU                  ::=   SET {
                                   [0] CHOICE {
                                        nonReceipt [0]  IMPLICIT NonReceiptInformation,
                                        receipt [1]  IMPLICIT ReceiptInformation},
                                   reported     IPMessageId,
                                   actualRecipient      [1]  IMPLICIT ORDescriptor OPTIONAL,
                                   intendedRecipient  [2]  IMPLICIT ORDescriptor OPTIONAL,
                                        -- only present if not actual recipient
                                        -- the O/R descriptor must contain an O/R name
                                   converted   P1.EncodedInformationTypes OPTIONAL}

NonReceiptInformation     ::=   SET {
                                   reason [0] IMPLICIT
                                        INTEGER {uaeInitiatedDiscard(0),
                                   autoForwarded(1)},
                                   nonReceiptQualifier [1] IMPLICIT
                                        INTEGER {expired(0), obsoleted(1),
                                             subscriptionTerminated(2)} OPTIONAL,
                                   comments [2] IMPLICIT PrintableString OPTIONAL,
                                        -- on auto-forward
                                   returned [3] IMPLICIT IM-UAPDU OPTIONAL}

ReceiptInformation        ::=   SET {
                                   receipt [0] IMPLICIT Time,
                                   typeOfReceipt [1] IMPLICIT INTEGER {
                                        explicit(0),automatic(1)} DEFAULT explicit,
                                   [2] IMPLICIT P1.SupplementaryInformation OPTIONAL}

END -- of P2 definitions
```

4.2.2 Common Data Types

The following two data types figure in the definition of both types of UAPDU.

4.2.2.1 User Message Id

A UserMessageId identifies a particular User Message and is used in providing the User Message Identification service. It is intended to be a unique and unambiguous identifier, and is assumed to be specified as recommended in the service definition.

A UserMessageId contains a Printable String and may also include an O/R name. If present, the O/R name denotes the UA that generated the identifier. The Printable String is assigned by the generating UA (or its user) and is intended to be unique within the context of that user. When the user's O/R name is present, therefore, the identifier is globally unique:

```
UserMessageId ::= [APPLICATION 11] IMPLICIT SET {
   ORName OPTIONAL,
   PrintableString}

ORName ::= P1.ORName
```

4.2.2.2 Time

A Time identifies a calender date and time of day to be conveyed in the protocol. It must always include the time of day in terms of UTC (Universal Coordinated Time), and may optionally convey the local time of the UAE that generates it. The precision of the time is to either one second or one minute, as selected by the generating UAE:

```
Time :: = UTCTime
```

4.2.2.3 O/R Descriptor

An ORDescriptor, can include an O/R name, a free-form name, and a telephone number. The O/R name is that needed to identify the user to the MTS. There are circumstances, involving the provision of certain services, in which the O/R name may not be omitted. Additionally or alternatively, a free-form name denoting the user may be included. This component allows inclusion, for example, of a familiar, user-friendly name not subject to the constraints imposed upon O/R names. It is intended to be of limited length, and is not intended to carry arbitrary information. The O/R descriptor may also include a telephone number that can be used, for example, to contact the user:

```
ORDescriptor :: = SET {-- at least one of the first two members must be present
   ORName OPTIONAL,
   freeformName [0] IMPLICIT T61String OPTIONAL,
   telephoneNumber [1] IMPLICIT PrintableString OPTIONAL}
```

4.2.3 User Message UAPDU

The User MessageUAPDU (UM-UAPDU) represents a User-Message. A UM-UAPDU has two parts, its heading and body:

```
UM-UAPDU :: = SEQUENCE {Heading, Body}
```

4.2.3.1 Heading

The Heading of a UM-UAPDU contains a number of components, each used in the provision of one of the UAS services defined in Section 3:

```
Heading :: = SET {
    UserMessageId,
    originator              [0]   IMPLICIT ORDescriptor OPTIONAL,
    authorizingUsers        [1]   IMPLICIT SEQUENCE OF ORDescriptor OPTIONAL,
                                  -- only if not the originator
    primaryRecipients       [2]   IMPLICIT SEQUENCE OF Recipient OPTIONAL,
    copyRecipients          [3]   IMPLICIT SEQUENCE OF Recipient OPTIONAL,
    blindCopyRecipients     [4]   IMPLICIT SEQUENCE OF Recipient OPTIONAL,
    inReplyTo               [5]   IMPLICIT UserMessageId OPTIONAL,
                                  -- omitted if not in reply to a previous message
    obsoletes               [6]   IMPLICIT SEQUENCE OF UserMessageId OPTIONAL,
    crossReferences         [7]   IMPLICIT SEQUENCE OF UserMessageId OPTIONAL,
    subject                 [8]   CHOICE {T61String} OPTIONAL,
    expiryDate              [9]   IMPLICIT Time OPTIONAL, -- if omitted, expiry date is never
    replyBy                [10]   IMPLICIT Time OPTIONAL,
    replyToUsers           [11]   IMPLICIT SEQUENCE OF ORDescriptor OPTIONAL,
                                  -- each O/R descriptor must contain an O/R name
    importance             [12]   IMPLICIT INTEGER {low(0), normal(1), high(2)} DEFAULT
    normal,
    sensitivity            [13]   IMPLICIT INTEGER {personal(1), private(2),
    companyConfidential(3)}
                                  OPTIONAL,
    autoforwarded          [14]   IMPLICIT BOOLEAN DEFAULT FALSE,
                                  -- indicates that the forwarded message body part(s)
                                  -- were auto-forwarded
preparationDate        [15]   IMPLICIT Time OPTIONAL,
warningDate            [16]   IMPLICIT Time OPTIONAL}
```

The originator component identifies the user who submitted the User-Message. The authorizingUsers component identifies the users who authorized the sending of the user-message.

The O/R Descriptor, see clause 4.2.2.3

The primaryRecipients, copyRecipients, and blindCopyRecipients are the users whom the originator indicated should receive the User Message in the indicated capacities.

A Recipient data element, used to represent the members of the above-mentioned components, identifies a recipient by means of an O/R descriptor. It also includes two per-recipent requests relevant to that recipient: reportRequest and replyRequest. The reportRequest data element can indicate requests for receiptNotification, nonReceiptNotification, and return of the User Message with a non-receipt notification (returnUserMessage). When receiptNotification is selected, nonReceiptNotification shall also be selected. The replyRequest data element denotes a request to the recipient that he reply to the User Message. If any of these per-recipient requests is made, the O/R descriptor must include an O/R name component so that the request can be acted upon:

```
Recipient :: = SET {
    [0] IMPLICIT ORDescriptor,
    reportRequest [1] IMPLICIT BIT STRING {
        receiptNotification(0), nonReceiptNotification(1), returnUserMessage(2)} DEFAULT {},
            -- if requested, the O/R descriptor must contain an O/R name
    replyRequest [2] IMPLICIT BOOLEAN DEFAULT FALSE
        -- if true, the O/R descriptor must contain an O/R name --}
```

The inReplyTo component identifies a previous User Message to which this one is a reply.

The obsoletes and crossReferences components identify any User-Messages obsoleted or cross-referenced, respectively, by this User-Message.

The subject component conveys the originator's indication of the subject of the User Message.

The expiryDate component indicates the date and time by which the originator considers the User Message to be no longer valid and useful. The time by which the originator would like to have a reply sent is specified as the replyByTime. The replyToUsers are those to whom the originator wishes any replies to be directed. The O/R descriptor of each such user must include an O/R name component so that the reply can be sent by means of the MTS.

The importance component is the originator's indication of the User-Message's importance. The importance may be low, normal, or high. The sensitivity component is the originator's indication of the User Message's sensitivity. The sensitivity may be personal, private, or companyConfidential.

If the body of the User Message contains another User Message that was autoforwarded, this is indicated.

The preparationDate component indicates the date and time at which the information constituting the body of the User Message was prepared.

The warningDate component indicates the date and time, prior to the expiry date, at which the originator suggests that the recipient might wish to be reminded of the User Message by his UA.

4.2.3.2 Body

The body of a UM-UAPDU comprises one or more independent parts:

```
Body :: = SEQUENCE OF BodyPart
```

Each part carries with it an indication of its type. Where necessary the representation of a particular body part type is structured to include, along with the actual body part information, a set of parameters which are associated with or required for the interpretation of the body part information. The various parts of a particular User Message's body may be of different types:

```
BodyPart :: = CHOICE {
        [0]  IMPLICIT IA5Text,
        [1]  IMPLICIT TLX
        [2]  IMPLICIT Voice,
        [3]  IMPLICIT G3Fax,
        [4]  IMPLICIT TIF0,
        [5]  IMPLICIT TTX,
        [6]  IMPLICIT Videotex,
        [7]  NationallyDefined,
        [8]  IMPLICIT Encrypted,
        [9]  IMPLICIT ForwardedUserMessage,
       [10]  IMPLICIT SFD,
       [11]  IMPLICIT TIF1,
       [12]  IMPLICIT ODIF,
       [13]  IMPLICIT Transparent,
       [14]  IMPLICIT ISO6937Text}
```

An IA5Text body part contains a string of characters, together with an indication of which repertoire (ITA2 or IA5) is in use. In either case, the characters are represented using the appropriate IA5 bit combinations:

```
IA5Text :: = SEQUENCE {
   SET {repertoire [0] IMPLICIT INTEGER {ia5(5), ita2(2)} DEFAULT ia5
       -- additional members of this Set are a possible future extension --},
       IA5String}
```

A ForwardedUserMessage body part represents a User Message which is contained in another User Message body for the purpose of conveying it to a further set of recipients. It includes, optionally, the delivery information supplied when the User Message was originally delivered:

```
ForwardedUserMessage :: = SEQUENCE {
   SET {
       delivery [0] IMPLICIT Time OPTIONAL,
       [1] IMPLICIT DeliveryInformation OPTIONAL}
       UM-UAPDU}
```

```
DeliveryInformation :: = SET {
        [0] IMPLICIT P1.ContentType,
        originator P1.ORName,
        original [1] IMPLICIT P1.EncodedInformationTypes,
        P1.Priority DEFAULT normal,
        [2] IMPLICIT DeliveryFlags,
        otherRecipients [3] IMPLICIT SEQUENCE OF P1.ORName OPTIONAL,
        thisRecipient [4] IMPLICIT P1.ORName,
        intendedRecipient [5] IMPLICIT P1.ORName OPTIONAL,
        converted [6] IMPLICIT P1.EncodedInformationTypes OPTIONAL,
        submission [7] IMPLICIT P1.Time}

ODIF :: = SEQUENCE OF ODIFProtocolElement -- see ECMA-xx
```

A Transparent body part contains a stream of bits structured in octets representing any kind of information:

```
Transparent :: = ANY

ISO6937Text :: = SEQUENCE {
   SET { repetoire[0] IMPLICIT INTEGER {part1and2 (0)} DEFAULT part1and2}
       SEQUENCE OF ISO6937Line}

ISO6937Line :: = [0] IMPLICIT OCTET STRING
```

The characters and commands allowed, their graphical symbols and 8-bit coded representations are those specified by ISO 6937, Part 1 and 2.

For the Body Types not explained above refer to the definitions in the CCITT Rec. X.420.

4.2.4 Status Report UAPDU

The Status Report UAPDU (SR-UAPDU) represents a notification of the receipt or non-receipt of a user-message:

```
SR-UAPDU :: = SET {
    [0] CHOICE {
        nonReceipt        [0] IMPLICIT NonReceiptInformation,
        receipt           [1] IMPLICIT ReceiptInformation},
    reported              UserMessageId,
    actualRecipient       [1] IMPLICIT ORDescriptor OPTIONAL,
    intendedRecipient     [2] IMPLICIT ORDescriptor OPTIONAL,
        -- only present if not actual recipient. The O/R Descriptor must contain an
        -- O/R name
    converted             P1.EncodedInformationTypes OPTIONAL}
```

A NonReceiptInformation component is included if, and only if, notification of non-receipt is being conveyed. Its components include a reason indicating either uaeInitiatedDiscard (if the UAE discarded the User Message for local reasons) or autoforwarded (if the UAE autoforwarded the User Message to another UAE). In the case of UAE discard, the nonReceiptQualifier conveys an indication of expired (if discard occurred due to arrival of the expiry date), obsoleted (if the user-message was obsoleted by another one), or subscriptionTerminated. In the case of autoforwarding, the comments component optionally conveys a string of characters, supplied by the UAE that performed the autoforwarding. If the originator requested return of the User Message upon non-receipt, the returned component conveys the entire UM-UAPDU that was delivered:

```
NonReceiptInformation :: = SET {
    reason                  [0] IMPLICIT INTEGER {uaeInitiatedDiscard(0), autoForwarded(1)},
    nonReceiptQualifier     [1] IMPLICIT INTEGER{expired(0), obsoleted(1),
                            subscriptionTerminated(2)} OPTIONAL,
    comments                [2] IMPLICIT PrintableString OPTIONAL, -- on auto-forward
    returned                [3] IMPLICIT UM-UAPDU OPTIONAL}
```

A ReceiptInformation component is included if, and only if, notification of receipt is being conveyed. Its receipt Time component indicates the time at which the UM-UAPDU was actually received. The typeOfReceipt will indicate either explicit (in case the recipient voluntarily and explicitly authorized the sending of the notification of receipt) or automatic (in case the UAE automatically generated the notification when the User Message was received). The use of SupplementaryInformation in the SR-UAPDU is reserved for the communication of responses from telex and telematic terminals (for example, answerback):

```
ReceiptInformation :: = SET {
    receipt [0] IMPLICIT Time,
    typeOfReceipt [1] IMPLICIT INTEGER {explicit(0), automatic(1)} DEFAULT explicit,
    [2] IMPLICIT P1.SupplementaryInformation OPTIONAL}
```

The reported UserMessageId identifies the User Message whose fate is being reported by this SR-UAPDU.

The actualRecipient component identifies the UAE to which the UM-UAPDU was delivered, and which is generating this status report. If the UM-UAPDU was delivered to the originally specified recipient, this component conveys exactly the corresponding O/R descriptor from the Recipient component that identified the actual recipient in the delivered UM-UAPDU. If the UM-UAPDU was delivered to an alternate recipient, this component is an O/R descriptor that identifies the actual recipient. It must, however, include the actual recipient's O/R Name.

The intendedRecipient component is returned only if the UM-UAPDU was actually delivered to an alternate recipient. This component is then the O/R descriptor that identifies the originally intended recipient, exactly as conveyed in the corresponding Recipient component of the UM-UAPDU being reported on.

The converted EncodedInformationTypes component conveys the new EncodedInformationTypes if the UM-UAPDU was subjected to conversion.

5. CONFORMANCE REQUIREMENTS

5.1 General

This Section defines the conformance requirements for the Message Interchange Protocols defined in Section 4 of this Standard. The requirement to support elements of the UAS and MTS protocols as defined in Section 4, depends on the services as defined in Section 3, as follows:

All protocols elements corresponding to the services described as "Fundamental" shall be supported by a MIDA system.

All protocol elements corresponding to service elements described as "Basic" shall be correctly handled by a MIDA system.Those described as "Basic for sending" shall be creatable by a MIDA system. Those described as "Basic for reception" shall be rendered by a receipient MIDA system. All shall be relayed by a relaying MIDA system.

All protocol elements supporting service elements described as "Optional" shall not be mandatory for creating or rendition. However, a relaying MIDA system shall relay these protocol elements unchanged.

Only the externally visible and externally testable criteria are specified.

5.2 Equipment

The conformance requirement is for equipment which consists of hardware and/or software claimed to be in conformance with this Standard. The equipment may also have other purposes.

5.3 Peer Equipment

Any execution of the protocols 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 this 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 conformance status between the two equipments.

5.4 Protocol Subsets

No protocol subsets are defined by this Standard.

5.5 Additional Message Interchange Protocols

In addition to the protocols defined in this Standard, the equipment may also implement other message interchange protocols.

Such additional provisions are themselves not in conformance with this Standard, but do not prejudice conformance with this Standard provided that they are separate and do not prevent use of the protocols defined within this Standard.

5.6 Requirements

As a minimum each equipment, conforming to this Standard, shall conform to the syntax and semantics of the MTS protocol.

A MIDA system may embody an RTS mapping onto the Basic Activity Subset Session Service (ISO 8326) or onto Presentation services ECMA-84, or both. The conformance statement shall specify which mappings the MIDA system supports.

A MIDA system need not embody User Agents to conform to this Standard. If User Agents are embodied, support of the UAS protocol must be specifically stated in the conformance statement.

Automatic routing of messages need not to be provided for all User Name Attributes defined in this Standard. The conformance statement must specify which User Name Attributes will support automatic routing. All MIDA systems must be capable of automatic routing on the basis of the O/R Address, (see clause 2.4.1.5).

Conformance to the UAS Protocol

The subject equipment:

- shall accept all correct messages received from peer equipment;

- shall only generate correct messages to peer equipment.

The conformance statement shall specify which (if any) of the service elements not classified as "Basic for sending" it can create, and which (if any) of the service element not classified as "Basic for reception" it can render.

Conformance to the MTS Protocol

The subject equipment:

- shall accept all messages received from peer equipment and react correctly to correct protocol elements;

- shall relay correct protocol elements in accordance with this Standard, and any unrecognised protocol elements unchanged to peer equipment;

- shall only generate correct messages to peer equipment.

The conformance statement shall specify which (if any) of the Message Transfer Service elements, defined as "Optional", are supported.

APPENDIX A

BRIEF DESCRIPTION OF THE REFERENCE MODEL OF OPEN SYSTEMS INTERCONNECTION

A.1 Introduction and Scope

This Appendix is not part of the Standard.

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 transferrring 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 interconnection is not only concerned with the transfer of information between systems (i.e. with communication), 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 contituents (see Figure A. 1):

- application processes within an OSI environment;

- connections which permit information exchange;

- 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, minitoring 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.

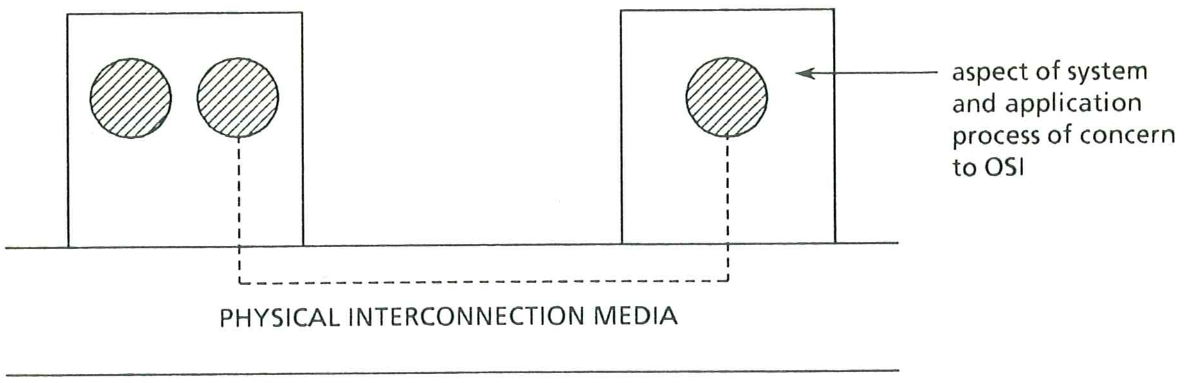

aspect of system and application process of concern to OSI
PHYSICAL INTERCONNECTION MEDIA

Figure A.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 Figure A.2.

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 specifieable 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.

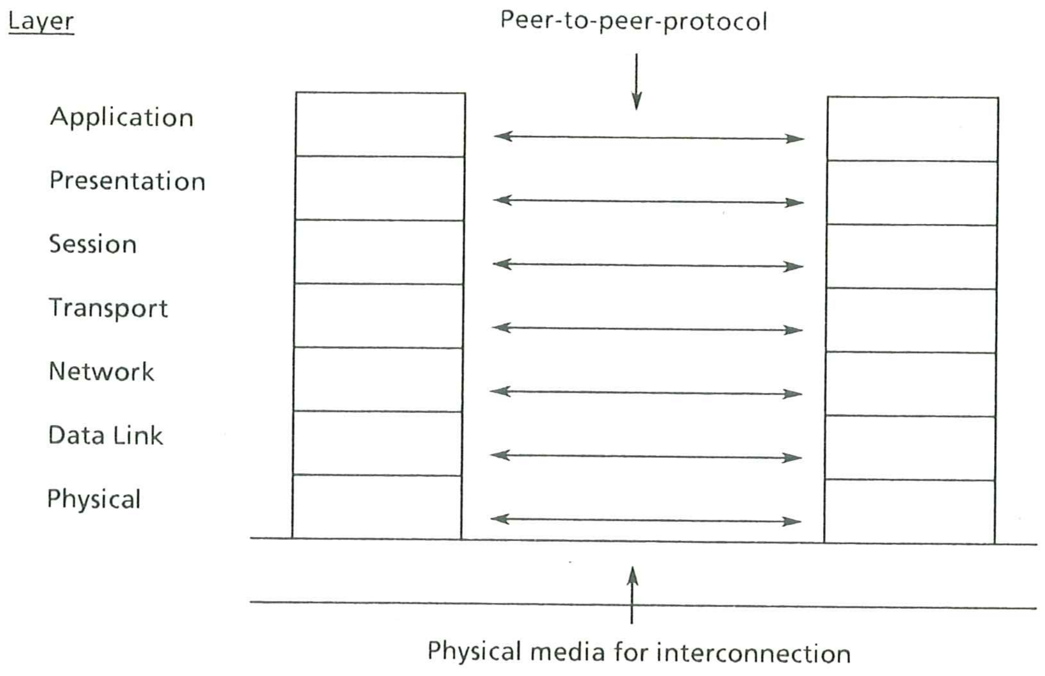


Figure A.2. The seven-layer Reference Model and peer-to peer protocol

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.

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

- external attributes of the virtual resource and its manipulation funtions exist in the Application Layer;

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

- 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 connections 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

GLOSSARY

B.1 Introduction and Scope

This Appendix is part of the Standard.

The terminology used in this Standard consists of :

- Terminology defined in the Reference Model for Open System Interconnection, (ISO 7498).
- Terminology for message interchange architecture, services and protocol, which is defined in this Appendix (see B.2.).
- Notational terminology, which is defined in Appendix C.

B.2 Definitions

Body
The information unit which the user requires to be transferred.

Communication Service
A service utilized by RTS which corresponds to levels 1-6 of the OSI model.

Envelope
The field of the message which carries the MTS protocol elements.

Message
The information unit exchanged between MTAs, including the Envelope and the User Message.

Message Handling System (MHS)
A system providing the fundamental "store and forward" service.

Distributed Application for Message Interchange (MIDA)
An application which provides the MHS(s) interconnection.

MIDA system
An MHS that can be interconnected with other MHS(s) utilizing the Distributed Application for Message Interchange.A MIDA system is composed of three kinds of functional entities : the UA (s) and the MTA (s), and the Reliable Transfer Service entities.

MIDA user
A human operator or a computer process using the MIDA system services.

Message Transfer Agent (MTA)
The functional entity concerned with "store and forward" features. The MTA interacts with the originator UA to which it offers the message delivery service. In addition, the MTA communicates with other MTA(s) via the MTS protocol.

Message Tranfer Agent Entity (MTAE)
The Protocol engine within a Message Transfer Agent.

Message Transfer Sublayer (MTS)
The middle sublayer in the MIDA system architecture, containing the MTA functional entities.

O/R name
The name of a user of the MIDA system, may be the same as the name of his UA, or constructor from real world parameters.

Originator
A MIDA user from whom the MIDA system accepts a body and sending attributes.

Originator UA
A UA submitting to the MTS a User Message to be transferred.

Protocol Engine
The functional entity which assembles or disassembles Protocol Data Units, and creates, or responds to, elements of protocol.

Recipient
A MIDA user who receives a body and receiving attributes from the MIDA system.

Recipient UA
A UA to which a User Message is delivered or that is specified for delivery.

Reliable Transfer Service (RTS)
A MIDA service, utilized by the MTA, providing the message transfer from one MTA to another. RTS lies in the Application Layer of the OSI model.

User Agent (UA)
The functional entity concerned with user visible features of the MIDA system. These features essentially are the User Message sending and the User Message receiving. Typically, the UA is a set of computer processes used for the creation and the manipulation of the User Messages.

The UA interacts with the originator or the recipient via an input/output device, requiring from the originator the control information ensuring the correct message handling. Secondly, the UA interacts with the MTA, in the layer below, co-operating for the envelope creation.

User Agent Entity (UAE)
The protocol engine within a UA.

User Agent Sublayer (UAS)
The upper sublayer in the MIDA system architecture, containing the UA functional entities.

User Message
The information unit exchanged between one originator and one or more recipients, together with the sending attributes.

B.3 Acronyms

ADMD	= Administration Management Domain
BR	= Basic Receiving
BS	= Basic Sending
BSR	= Basic Sending and Receiving
DP-A	= Data Presentation A
MD	= Management Domain
MHS	= Message Handling System
MPDU	= Message Protocol Data Unit
MT	= Message Transfer
MTA	= Message Transfer Agent
MTAE	= Message Transfer Agent Entity
MTS	= Message Transfer Sublayer
ODIF	= Office Document Interchange Format
O/R	= Originator/Recipient
OSI	= Open Systems Interconnection
PRMD	= Private Management Domain
P1	= Message Transfer Sublayer Protocol
P2	= User Agent Sublayer Protocol
RSDU	= Reliable Transfer Service Data Unit
RT	= Reliable Transfer
RTS	= Reliable Transfer Service
SDE	= Submission and Delivery Entity
SMPDU	= Service Message Protocol Data Unit
SPDU	= Session Service Data Unit
SSAP	= Session Service Access Point
TSAP	= Transport Service Access Point
UMPDU	= User Message Protocol Data Unit
UA	= User Agent
UAE	= User Agent Entity
UAS	= User Agent Sublayer
UTC	= Universal Coordinate Time

APPENDIX C

NOTATION

C.1 Introduction and Scope

This Appendix is part of the Standard.

C.2 Definitions

The sublayer services described for MIDA relate to the MTS and the RTS.

In these definitions "N" sublayer can refer to Message Transfer Sublayer = "MT" or Reliable Transfer Service = "RT".

"X" service
A conceptual unit of the "N" sublayer service, of which "X" is a particular name.

[service] primitive
A discrete component of an "X" - service.

X.request primitive
A type of primitive by which a "N" sublayer user causes an occurrence of the "X"- service.

X.confirmation primitive
A type of primitive by which a "N" sublayer user is informed of another "N" sublayer user´s acceptance of a previously issued X.request.

X.indication primitive
A type of primitive by which a "N" sublayer user is informed of another "N" sublayer user´s previously issued X.request.

X.response primitive
A type of primitive by which a"N" sublayer user accepts an X.indication.

Event
The occurrence of a primitive.

Initiator
The "N" sublayer user which issues the "request" primitive of the "X"-service concerned.

Responder
The "N" sublayer user which receives the "indication" primitive of the "X"-service concerned.

C.3 Service Model

The service is modelled as an abstract service to which MIDA service users (M-users) gain access at MIDA service access points (MSAPs). These MSAPs are uniquely identified by MIDA service addresses.

Provision of the MIDA service may involve interactions among MIDA service provider entities of the MIDA layer. It may involve interactions among one or more MIDA service users located at separate MSAPs.

C.4 Primitives

The MIDA service is defined by means of service primitives. Primitives are conceptual, and are not intended to be directly related to 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 M-user and an implementation are not defined.

The subdivision of the MIDA service into the particular primitives chosen is arbitrary, in that the same MIDA service could be described by other logically equivalent primitives.

A primitive occurs at one MSAP. It usually has parameters containing values relating to its occurrence.

The occurrence of a primitive is logically instantaneous and indivisible event. The primitive occurs at a logically separate instant. It occurs either completely or not at all.

The primitives are given names prefixed by "M" (MIDA) to distinguish them from primitives of adjacent-layers. The names of primitives are written in upper case, e.g. MT-DELIVER.

There are four types of primitives in this Standard :

- request primitive
- indication primitive
- response primitive
- confirmation primitive

Request
A request is issued by a service user to invoke some procedure.

Indication
An indication is issued by a service provider to either invoke some procedure or indicate that a procedure has been invoked by the service user at a peer service access point.

Response
A response is issued by a service user to complete at a particular service access point some procedure previously invoked by an indication at that service access point.

Confirmation
A confirmation is issued by a service provider to complete at a particular service access point some procedure previously invoked by a request at that service access point.

C.5 Service Structure

Each service consists of one or more primitives and affects one or more MSAP. The main service structure used in this Standard illustrates the allowed sequence of event at a particular service access point.

Figures C.1, C.2 and C.4 are time-sequence diagrams giving a picturial representation of the service structures. The progression of time is in the vertical axis downwards. The two sides represent the two M-service users. The void in the middle represents the MIDA service provider. The arrows give the direction of event propagation.

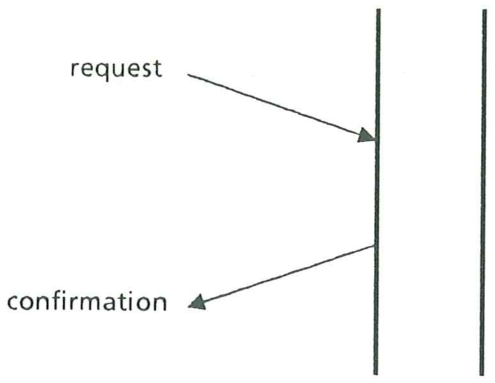


Figure C.1

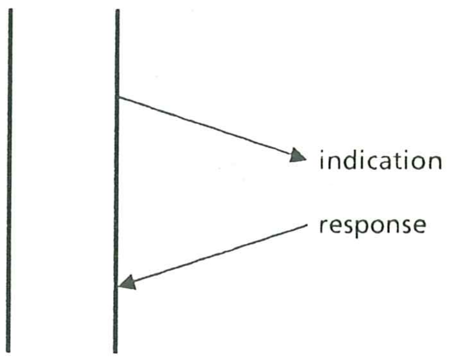


Figure C.2

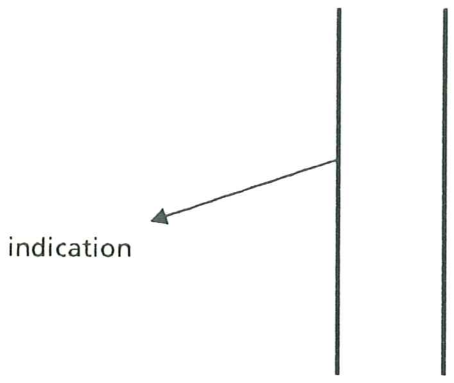


Figure C.3

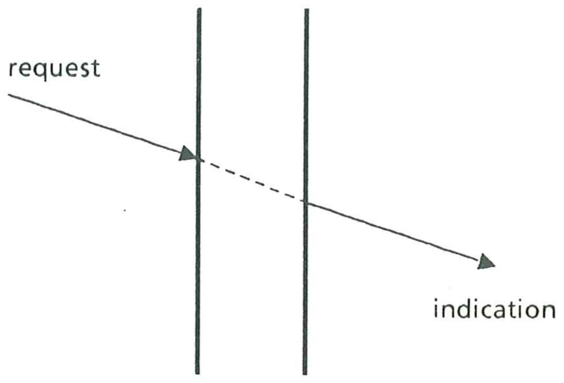


Figure C.4

Unlike the events which are the occurrence of its constituent primitives, the occurrence of a service structure is not logically instantaneous and indivisible. The intervals between its constituent events may be non disruptively interspersed with other events.

C.6 Effects of Services

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

The effects of a service are referred to as being expedited if it´s indication or confirmation primitives may arrive at the other service access point before those of a previous occurrence of a service.

The effects of a service are referred to as being disruptive if it may destroy, and therefore prevent the occurrence of, indication and confirmation primitives corresponding to previous request or response primitives.

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

C.7 Parameter Notation

For each service structure the parameters are defined by a parameter table and for each service primitive by a parameter description.

Parameter Table
The vertical axis of each table is a list of the parameters. The column heading indicates the primitive types:

Req	for request.
Ind	for indication.
Resp	for response.
Conf	for confirmation.

Parameter description
It contains a detailed description of parameters associated with each service primitive. For each parameter it includes, its purpose and, if necessary, notes for additional information.

Type

Each parameter is classified as one of the following two types :

Mandatory (M) :	A parameter that must always be supplied (although the value may be null or defaulted where appropriate).
Conditional (C) :	A parameter that is present under certain conditions.

APPENDIX D

SUMMARY OF PRESENTATION TRANSFER SYNTAX

D.1 Introduction and Scope

This Appendix is not part of the Standard.

The protocol specified by this Standard is based upon the transfer syntax defined in the CCITT Rec. X.409. This Appendix describes briefly that syntax and some of the associated concepts.

D.2 Data Types, Data Values and Standard Notation

Rec. X.409 defines a transfer syntax for various kinds of information. Each piece of information is considered to have a type as well as a value. A data type is a class of information (for example, numeric or textual). A data value is an instance of such a class (for example, a particular number or a fragment of text). Rec. X.409 defines a number of generally useful data types from which application-specific data types are constructed in this Standard and in others that make use of the Rec. X.409 transfers syntax. Among the generally useful data types defined by Rec. X.409 are Integer, Octet, String, Sequence and Set.

The standard notation defined in Rec. X.409 is a formal description method that allows data types relevant for an application to be specified in terms of other data types, including the gererally useful data types of Rec. X.409. This notation is used in section 4 of this Standard, where the Protocol Element data types are specified in terms of Sets and Sequences of more elementary data types which in turn are specified in terms of others, and finally in terms of basic data types such as Integer and Octet String.

D.3 Standard Representation

The standard representation for a data type is the set of rules for encoding values of that type for transmission as a sequence of octets. The representation of a value also encodes its type and length, and is completely implied by the standard notation of the data type.

The standard representation of a data value is a data element having three components, which always appear in the following order: The Identifier designates the data type and governs the interpretation of the Contents. The Length specifies the length of the Contents. The Contents is the substance of the object, containing the primary information the object is intended to convey. The Identifier and the Length each consist of one or more octets; the Contents consists of zero or more octets.

D.3.1 Identifier

Four classes of data types are distinguished by means of the Identifier: universal,application-wide, context-specific and private-use. Universal types are generally useful, application-independent types; they are defined in Rec. X.409. Application-wide types are more specialized, being peculiar to a particular application; they are defined in this Standard and in others using Rec. X.409, by means of the standard notation. Context-specific types, like application-wide types, are peculiar to an application and defined using the standard notation. However, they are used only within an even more limited context - for example, that of a Set - and their identifiers are assigned so as to be distinct only within that limited context. Private-use types are reserved for private use; the assignment of specific private-use Identifiers can be accomplished by means of the standard notation but is outside the scope of Rec. X.409 or this Standard.

Two forms of data elements are distinguished by means of the Identifier: Primitive and constructor. A primitive element is one the Contents of which is atomic. A constructor element is one the Contents of which is itself a data element, or a series of data elements. Constructor elements are thus recursively defined.

D.3.2 Length

The Length specifies the length in octets L of the Contents and is itself variable in length. It may take any of three forms; short, long and indefinite.

- The short form is used when L is less than 128.
- The long form is used when L is greater than 127.
- The indefinite form may (but need not) be used when the element is a constructor. When this form is employed, a special end-of-contents (EOC) element terminates the Contents.

D.3.3 Contents

The Contents is variable in length and is interpreted in a type-dependent way. If the data element is a constructor, the Contents itself comprises zero or more elements; data elements are thus recursively defined.

D.4 Built-in Types and Defined Types

The generally useful data types defined by Rec. X.409 consist of built-in types and defined types.

Built-in types are used to construct all other data types. They include Integer, Octet String, Sequence, Set and Tagged. Integer is a primitive data type. Octet String can be either primitive or constructor. Sequence and Set are constructor data types. Identifiers for these data types are of the universal class and are specified in Rec. X.409. A Tagged data type is a data type for which the Identifier can be specified using the standard notation, as is done in section 4 of this Standard.

Defined types are specified in Rec. X.409 using the standard notation. They include Numeric String and Printable String, all of which are defined in terms of the built-in type Octet String. They can be either primitive or constructor; their identifiers are of the universal class and are specified in Rec. X.409.

APPENDIX E

RELATIONSHIP OF MIDA SERVICES AND PROTOCOLS TO CCITT's MHS SERVICES AND PROTOCOLS

E.1 Introduction and Scope

This Appendix is not part of the Standard.

This Appendix has been written especially to facilitate the comparison with the CCITT Recommendations, especially for readers who are already familiar with the CCITT MHS.

E.2 Model

The model used by MIDA is very similar to the CCITT model. However, the concept of Submission and Delivery Entities (SDE) is not yet addressed in the MIDA Standard and is for further study. This also means that there is no Submission and Delivery Protocol (P3). The only other differences are in the view on sublayers, where MIDA explicitly refers to the Message Transfer and User Agent "Sublayers" and also refers to a "User Sublayer" which is not present in the CCITT model. Another difference is that CCITT uses the expression "IP Message" (Interpersonal) which is referred to as "User Message" in MIDA.

E.2.1 Addressing

Due to the absence of a standard for directories, MIDA has defined an O/R address based on Architectural Characteristics of the Message Transfer System.

In a MIDA system the only mandatory form of addressing is the O/R Address which is built up as follows:

- Country Code
- Administration Domain Name
- Private Domain Name
- MTA Name
- UA Local Identifier

However all the forms specified by CCITT can be used as options.

For more details, see clause 2.4.1.5 of the Standard.

In the clause on construction of O/R Names in Rec. X.400, four categories of standard attributes are described. To the example of Architectual Attributes MIDA has added the "MTA Name" and the "UA Local Identifier".

The fields MTAName and UALocalId have been included in the P1 Protocol using the DomainDefinedAttribute parameter. This is no explicit change to the Protocol, just a matter of using it.
See clause 4.1.5.1 of the Standard.

In connection with the MPDUIdentifier, in the P1 Protocol , MIDA uses the GlobalDomainIdentifier field to store the MTAName.
See clause 4.1.5.6 of the Standard.

E.3 Services

The classification of services are slightly different in that:

"Basic" in MIDA = "Basic" + "Essential Optional" in CCITT

"Optional" in MIDA = "Additional Optional" in CCITT

In MIDA there exists a subgroup of services called "Fundamental" . The corresponding services exist also in CCITT, but are implied in the text rather than listed as separate services.

E.3.1 Message Transfer Service

The Message Transfer Service in MIDA are identical to the CCITT services with the following exceptions:

E.3.1.1 CCITT Services not included in MIDA

The following services deemed to be associated with the SDE - MTAE Communication are excluded:

- Access Management
- Registered Encoded Information Types
- Deferred Delivery Cancellation
- Alternate Recipient Assignment
- Hold for Delivery

These services will be considered for inclusion in the ongoing work on the MIDA General Access Service.

The Conversion services were deemed as not neccessary in MIDA and therefore the following two CCITT services are excluded:

- Explicit Conversion
- Implicit Conversion

Note, however that the services Conversion Prohibition and Converted Indication have been retained in order to facilitate the co-operation with CCITT Domains.

E.3.1.2 CCITT Service which is differently classified

- Deferred Delivery	See clause 3.1.1.2.3 of the Standard

This service is defined as "Essential Optional" in CCITT, but has been classified as "Optional" in MIDA. The reason is that the Deferred Delivery Service cannot be guaranteed in a Private Management Domain and therefore does not qualify as a Basic service.

E.3.1.3 MIDA Services which are not offered by CCITT

The following services are offered by MIDA as extentions to the CCITT services:

- Cancellation Time Selection (Optional)	See clause 3.1.1.2.10 of the Standard
- Originator-requested Alternate Recipient (Optional)	See clause 3.1.1.2.12 of the Standard
- Audit Trail (Basic)	See clause 3.1.1.4.2 of the Standard

E.3.2 User Agent Services

All the CCITT services are also included in MIDA.

The classification of Basic services has been extended to distinguish between Basic Receiving, Basic Sending and Basic Sending and Receiving in analogy with Rec. X.401.

The following two naming differences should be noted:

- The term "IP-message" in CCITT is replaced by "User Message" in MIDA. This is reflected in a few service names.
- The CCITT service "Typed Body" has been renamed "Typed Body Part" which more correctly describes the service as specified in the protocol.

E.3.2.1 MIDA Services which are not offered by CCITT

The following services are offered by MIDA as extentions to the CCITT services:

- Preparation Date (Optional)	See clause 3.2.1.6.1 of the Standard
- Warning Date (Optional)	See clause 3.2.1.6.2 of the Standard

The service Typed Body Part has been extended to include also the types:

- ODIF	See clause 3.2.1.4.2 of the Standard
- Transparent (Octet String)	
- ISO 6937 Text	

E.3.3 Reliable Transfer Service

The CCITT Reliable Transfer Service is based on the CCITT Rec. X.215 and Rec. X.225.

MIDA offers two alternative realizations:

- The Session Layer Services as specified in ISO DIS 8326 using the Basic Activity Subset (BAS). This is equivalent to the CCITT realization.
- The Presentation Layer Services as specified in ECMA-84 and the Basic Syncronized Subset (BSS) from the ISO 8326.

The first alternative is the default in MIDA.

In the S-CONNECT.request service primitive add the value 2 for MIDAp1 in the applicationProtocol field. See Appendix F, clause F.2.1.

E.4 Protocols

The protocols specified in this Standard are identical to the corresponding protocols in CCITT Rec. X.411 and Rec. X.420 on Message Handling Systems, with the exception of the differences detailed in this Appendix.

The MIDA protocols also use the presentation transfer syntax and notation, and the Remote Operations facilities and Reliable Transfer Server, specified in Rec. X.409 and Rec. X.410.

The clauses that follow detail the differences between CCITT's and ECMA's Message Transfer Protocols, P1; their Interpersonal Messaging (CCITT) and User Agent (ECMA) Protocols, P2. Each clause is organized on the basis of the service elements with which the protocol differences are associated. The differences are described in terms of the changes that would have to be made to the CCITT protocol to produce the corresponding ECMA protocol.

E.4.1 P1 Differences

Grade of Delivery Selection

1. Rename **deferrable** the **nonUrgent** distinguished value of Priority.

Cancellation Time Selection

2. Add the following member to the **UMPDUEnvelope** Set:

   latestDelivery [3] IMPLICIT Time OPTIONAL

3. Add **latestDeliveryTimePassed(3)** as a distinguished value of **ReasonCode**.

Delivered Info

4. Remove **DEFAULT public** from the **typeOfUA**.

Alternate Recipient Specified

5. Add the following member to the **RecipientInfo** Set:

   alternateRecipient [3] IMPLICIT ORName OPTIONAL -- none if absent

   Note E.1
   When **RecipientInfo** *appears in a* **ProbeEnvelope**, *the above member is omitted.*

   Note E.2
   The **recipient** *and* **intendedRecipient** *members of the* **ReportedRecipientInfo** *Set are used in connection with this, as well as the Alternate Recipient Allowed, service element.*

6. Add Deliver To Request information in bits 6-7 of the **Per-recipient Flag.**

Intended Recipient	= 00
Originator-specified Alternate Recipient	= 01
Domain-defined Alternate Recipient	= 10
Not used	= 11

Note E.3
Bits 5 to 7 are set to ZERO between Domains.
(For more details see clause 4.1.6.1 of the Standard)

Internal Trace Info

7. Add the following member to the **UMPDUEnvelope** , **DeliveryReportEnvelope** and **ProbeEnvelope** Sets:

```
InternalTraceInfo OPTIONAL

InternalTraceInfo :: = [ PRIVATE 0 ] IMPLICIT SEQUENCE OF PerMTATraceInfo

PerMTATraceInfo :: = SEQUENCE {
                         MTAName,
                         MTASuppliedInfo}

MTAName :: = PrintableString            -- Unique within a Management Domain

MTASuppliedInfo :: = SET {
                         arrival [0] IMPLICIT Time,
                         deferred [1] IMPLICIT Time OPTIONAL,
                         action [2] IMPLICIT INTEGER {relayed (0), rerouted (1)}
                         previous MTAName OPTIONAL}
```

8. Add the following member to the **DeliveryReportContent** Set:

```
intermediate InternalTraceInfo OPTIONAL
```

Conversion Prohibition

9. Remove the **ExplicitConversion** member from the **RecipientInfo** Set. The latter is referenced from both the **UMPDUEnvelope** and the **ProbeEnvelope**.

10. Delete the **ExplicitConversion** type definition.

Note E.4
In connection with the converted member of the **DomainSuppliedInfo** *Set, it should be noted that MIDA does not support encoded information type conversion of any kind. This Set is referenced indirectly from both the* **UMPDUEnvelope** *and the* **ProbeEnvelope**.

Original Encoded Information Types Indication

11. Add the following distinguished values to the Bit String that appears as a member of the **EncodedInformationTypes** Set:

```
oDIF(10) , iSO6937text(11)
```

O/R Name

12. Add the following member to the **PersonalName** Set:

```
title [4] IMPLICIT PrintableString OPTIONAL
```

13. Add the following members to the **Standard AttributeList** Sequence:

```
[7] IMPLICIT PositionorRole  OPTIONAL,
GeographicalAttributes  OPTIONAL

PositionorRole :: = PrintableString

GeographicalAttributes :: = [PRIVATE 1] IMPLICIT SET {
   streetNameNumber [0] IMPLICIT PrintableString OPTIONAL,
   townName [1] IMPLICIT PrintableString OPTIONAL,
   regionName [2] IMPLICIT PrintableString OPTIONAL}
```

14. Note that MIDA uses the Message Handling Address Variant 3, as below:

```
ORName:: = [APPLICATION 0] IMPLICIT SEQUENCE{

            SEQUENCE{
               CountryName,
               AdministrationDomianName,
               [2]  PrivateDomainName,
               [3]  IMPLICIT OrganisationName OPTIONAL,
               [5]  IMPLICIT PersonalName OPTIONAL,
               [6]   IMPLICIT SEQUENCE OF OrganisationalUnit OPTIONAL}

            SEQUENCE OF
            SEQUENCE{
               type PrintableString  =  "MIDA"
               value PrintableString  =  MTAName:UALocalId}}
```

E.4.1.1 Establishing and Releasing Associations

In CCITT a number of features in relation to associations are negotiated when opening an association or are subject of bilateral agreements. MIDA provides default values for these features as the number of private domains may be so large that negotiations and bilateral agreements are impracticable. Therefore the following features have been given default values:

I) Maximum number of associations:
1 in each direction

II) Monologue or Two Way Alternate:
Monologue

III) MTA responsible for establishing the association:
MTA with a message to send

IV) Permanent or Temporary association:
Temporary

V) Lower layer mapping:
To Basic Activity Subset Session services

VI) P1 Protocol:
CCITT-compatible

VII) Naming and Adressing, User Agent and Message Transfer:
Any of the CCITT-defined variants

VIII) RT-OPEN User-data:
Password omitted

Associations are established using the RT-OPEN.request primitive. The parameter Application-protocol can have the value "P1" to signify that the exact CCITT Protocol will be used or the value "MIDA P1" to signify that the extended MIDA Protocol will be used.
For more details see clause 4.1.9.1 of the Standard.

E.4.1.2 Lower Level Addressing

For details of the MIDA realization see clause 4.1.9.4 of the Standard.

E.4.2 P2 Differences

Dates

1. Add the following members to the **Heading** Set:

```
preparationDate [15] IMPLICIT Time OPTIONAL
warningDate [16] IMPLICIT Time OPTIONAL
```

Typed Body

2. Add the following alternatives to the **BodyPart** Choice:

```
[12] IMPLICIT ODIF
[13] IMPLICIT Transparent
[14] IMPLICIT ISO6937Text
```

3. Add the following three type definitions:

```
ODIF :: = SEQUENCE OF ODIFProtocolElement -- See ECMA-xx

Transparent :: = ANY

ISO6937Text :: = SEQUENCE {
   SET { repetoire[0] IMPLICIT INTEGER {part1and2 (0)} DEFAULT part1and2}
   SEQUENCE OF ISO6937Line}

ISO6937Line  :: = [0] IMPLICIT OCTET STRING
```

4. Throughout the **P2** module make the following substitutions:

```
- IM-UAPDU              --> UM-UAPDU
- IPMessageID           --> UserMessageId
- Forwarded IPMessage   --> Forwarded UserMessage
- return IPMessage      --> return UserMessage
```

APPENDIX F

REALIZATION OF RTS BASED ON PRESENTATION AND BAS SESSION SERVICES

F.1 Introduction and Scope

This Appendix, which is part of the Standard, describes how the Reliable Transfer Server (RTS) is supported by the OSI Presentation and BAS Session Services. The RTS has minimal requirements on the Presentation Layer itself. However, it makes extensive use of the Session Layer Services of ISO 8326, which are made directly available to Application Layer entities by the Presentation Layer. A minimal Presentation Layer protocol, which meets the needs of the RTS, is implicitly defined in F.2.1 below, and the remainder of this Appendix defines the use of the Session Layer Services by the RTS. The encoding of various parameters is defined using the notation of CCITT Rec. X.409.

The subset of the Session Service used is composed of the following functional units. The corresponding session services are also listed:

Kernel	Session Connection Normal Data Transfer Orderly Release U-Abort P-Abort
Exceptions	User Exception Reporting Provider Exception Reporting
Activity Management	Activity Start Activity Resume Activity End Activity Interrupt Activity Discard Please Tokens Give Tokens Give Control
Half-duplex	Give Tokens Please Tokens
Minor synchronize	Minor Synchronization Point Give Tokens Please Tokens

Note F.1
Bit Alignment. Where referenced in this Standard, bit positions in octets are as defined in Rec. X.409. Bit 8 as defined there corresponds to the high-order bit of the Session Service, and bit 1 corresponds to the low-order bit.

F.2 Session Connection Establishment Phase

The RTS attempts Session Connection Establishment in response to either of the following:

1. RT-OPEN: An RT-OPEN.request issued by the RTS-user.

2. RECOVER: An internal rule when attempting to recover a session connection that was aborted.

The two cases are distinguished by means of data elements in the SS-User Data parameter.

An RTS may reject an incoming connect indication, for example, because of RTS congestion. Specific reasons are outlined in the connect response.

F.2.1 S-CONNECT

Parameter	Req/Ind	Resp/Conf
Session Connection Identifier	1	1
Calling SSAP Address	2	
Called SSAP Address	2	2
Result	3	
Quality of Service (QOS)	4	4
Session Requirements	5	5
Synchronization Point Serial Number	6	6
Initial Data Token Assignment	7	7
Initial Minor Synch Token Assignment	7	7
Initial Major/Act Token Assignment	7	7
SS-User Data	8	8

1. The Initiating RTS will supply a Session Connection Identifier, which will be used to uniquely identify the connection. This identifier is formed of the following components: Calling SS-User Reference, Common Reference, and, optionally, Additional Reference Information. The identifier is returned unchanged by the responding RTS, except that the Calling SS-User Reference supplied by the initiator is conveyed as the Called SS-User Reference.

   Each component, when present, will contain a data element of the appropriately named type from the following definitions:

   CallingSS-UserReference :: = SSAPAddress --of the initiator

   CommonReference :: = UTCTime

   AdditionalReferenceInformation :: = T61 String

   The SSAPAddress is equivalent to the transport address (See 2 below) and is represented as a string of T.61 characters:

   SSAP :: = T.61 String

   The UTCTime field is used to differentiate between session connections initiated by the same RTS entity. It should not be used as a definitive statement of the connect request time.

2. Message handling will not use Session Layer addressing, that is, a session address will not be passed in the Connect SPDU of the Session Layer.

   The SSAP Addresses are passed down to (and then passed up from) the Transport Layer. They have a one-to-one relationship with transport addresses.

3. The receiving RTS may choose to ACCEPT or REJECT the S-CONNECT.indication. The Result parameter indicates which has occurred. Some qualification of the reason for the rejection may be present in the User Data parameter. The Session provider may also reject the connection request under some circumstances.

4. The parameters Extended Control and Optimized Dialogue Transfer are set to not required. The remaining parameters are set such that default values are used.

5. This parameter specifies the functional units to be used as listed in F.1 above.

6. Set to ZERO.

7. The connecting RTS will always request that the data token be available to obtain either a one-way monologue or two-way alternate dialogue session.

   In the RT-OPEN case, the connecting RTS will specify which RTS will initially hold the data token (minor synch token and major/activity token) upon successful completion of the connection phase, according to the initial-turn parameter supplied by means of RT-OPEN.request.

   The connecting RTS must assign all of the tokens to the same RTS. The connection may be rejected if this rule is violated. At any particular point in time, the holder of the tokens is referred to as the sending RTS, the other as the receiving RTS.

   In the RECOVER case, the following rules apply:

   - If the RTS has the tokens, it specifies that it retains them.

   - If the RTS does not have the tokens, but has issued an S-CONTROL-GIVE.request with no confirmation that the tokens were received, it gives the tokens to the other RTS. (Receipt of data serves as confirmation that the tokens were received.)

   - If the RTS does not have the tokens and does not have an S-CONTROL-GIVE.request outstanding, it specifies that the assignment of the tokens is acceptor chooses. In this case, the responding RTS will either keep or return the tokens depending upon whether it had them before the session was aborted.

8. In the S-CONNECT.request, this parameter contains a (Presentation Layer) PConnect data element. In the S-CONNECT.response, it contains a PAccept or PRefuse data element depending upon the Result parameter. These data elements are defined as follows:

```
PConnect :: = SET {
   [0] IMPLICIT DataTransferSyntax,
   [1] IMPLICIT pUserData SET {
      checkpointSize [0] IMPLICIT INTEGER DEFAULT 0,
      windowSize [1] IMPLICIT INTEGER DEFAULT 3,
      dialogueMode [2] IMPLICIT INTEGER {monologue(0), twa(1)} DEFAULT monologue,
      [3] ConnectionData,
      applicationProtocol [4] IMPLICIT INTEGER {p1(1), mIDAp1(2), p3(3)} DEFAULT p1}}

PAccept :: = SET {
   [0] IMPLICIT DataTransferSyntax,
   [1] IMPLICIT pUserData SET {
      checkpointSize [0] IMPLICIT INTEGER DEFAULT 0,
      windowSize [1] IMPLICIT INTEGER DEFAULT 3,
      [2] ConnectionData}}

PRefuse :: = SET {[0] IMPLICIT RefuseReason}

DataTransferSyntax :: = SET {[0] IMPLICIT INTEGER {x.409(0)}}
```

The only value defined for the DataTransferSyntax parameter refers to the Presentation Transfer Syntax specified in Rec. X.409 .

```
ConnectionData :: = CHOICE {
  open [0] ANY, -- RTS user data
  recover [1] IMPLICIT SessionConnectionIdentifier}

RefuseReason :: = INTEGER {
  rtsBusy(0), cannotRecover(1), validationFailure(2), unacceptableDialogueMode(3),
  unacceptableProtocol(4)}
```

The SessionConnectionIdentifier parameter is used to specify the previous session connection which was abnormally terminated. This is used in order to relate the new session to the existing RTS association.

The windowSize parameter allows negotiation of the number of outstanding minor synchronization points before data transfer must be suspended. The receiving RTS must supply a value in the Response or Confirmation that is less than or equal to the value requested. This becomes the agreed size and governs both directions of transfer.

The checkpointSize parameter allows negotiation of the maximum amount of data (in units of 1024 octets) that may be sent between two minor synchronization points. A value of zero indicates that no checkpointing will be done. The receiving RTS must supply a value in the Response or Confirmation that is less than or equal to the value requested. Exceptionally, it may supply any value in the case that the value of zero has been requested by the transmitting RTS. The value supplied by the receiving RTS becomes the agreed maximum value and governs both directions of transfer.

F.3 Data Transfer Phase: Actions of Sending RTS

Each RSDU, conveyed in an RT-TRANSFER.request, constitutes a Session Activity. For each session connection, only one activity may exist at a time, and only one interrupted activity awaiting resumption may exist at a time, including any that may have been aborted by S-ABORT.

An activity is transferred as a single SSDU if checkpointing is not used or the RSDU is smaller than the checkpointSize. Otherwise, the RSDU is transferred as a series of SSDUs, the maximum size of each being the negotiated checkpointSize.

F.3.1 Starting a New Activity

To start a new activity, the sending RTS (that which holds the tokens) issues an S-ACTIVITY-START.request. The sending RTS may start transmitting the APDU in an S-DATA.request immediately after the S-ACTIVITY-START.request is issued, since the latter service is not confirmed.

F.3.1.1 S-ACTIVITY-START (New RSDU)

Parameter	Req/Ind
Activity Identifier	1
SS-User Data	–

1. The Activity Identifier identifies the RSDU by means of a serial number. The first RSDU started on the session connection is assigned the number 1. Each successive RSDU for

that direction of transfer is assigned the next number. Thus numbering is separate for each direction of transfer.

F.3.2 Transmitting an RSDU

The maximum SSDU size, as stated previously, will have been negotiated during the connection phase. The sending RTS must submit in S-DATA.requests, SSDUs that conform to that agreement.

After each SSDU (S-DATA.request), the sending RTS normally inserts a checkpoint (S-SYNCH-MINOR.request) or indicates the end of the activity (S-ACTIVITY-END.request). In abnormal circumstances, such as upon receipt of an error report from the receiving RTS, the sending RTS may interrupt the activity (S-ACTIVITY-INTERRUPT.request), discard the activity (S-ACTIVITY_DISCARD.request), or even abort the connection (S-U-ABORT.request), as described in F.3.5.

Consecutive S-DATA.requests must not be given, and all data transfer must take place within an activity.

F.3.2.1 S-DATA

Parameter	Req/Ind
SS-User Data	1

1. The maximum size is as described above.

F.3.3 Insertion of Checkpoints

Checkpoints may only be inserted if a checkpointSize greater than zero was negotiated during the connection phase.

To facilitate checkpointing, the receiving RTS must be able to secure each SSDU. The sending RTS will interpret a confirmed checkpoint as signifying that the data has been secured and therefore does not require re-transmission.

To insert a checkpoint into the data stream, the sending RTS uses the minor synchronization service which has the following primitives and parameters.

F.3.3.1 S-SYNCH-MINOR

Parameter	Req/Ind	Resp/Conf
Type	1	
Synchronization Point Serial Number	2	2
SS-User Data	–	–

1. The RTS uses only the explicit confirmation expected type of minor synchronization. Each minor synchronization must be confirmed in the order received by the receiving RTS. When the sending RTS receives the confirmation to a minor synch, it assumes that the receiving RTS has secured the data up to that point. If the receiving RTS has detected a problem, it should not confirm the minor synchronization, but rather should issue an S-U-EXCEPTION-REPORT.request or an S-U-ABORT.request, depending upon the severity of the problem.

The sending RTS may issue further S-DATA.requests and S-SYNCH-MINOR.requests unless the agreed window size has been reached. The window is advanced when a Confirmation is received. This window is not policed by the Session Layer.

Note F.2
The minor synchronization window mechanism should not be used for flow control. The Transport Service provides a back-pressure flow control mechanism to control the rate of acceptance of data, in units of less than an SSDU.

2. The Session-provider allocates checkpoint serial numbers and passes them to the sending and receiving RTSs to associate with the transmitted data.

F.3.4 Normal Termination of an Activity

When an entire RSDU has been transmitted, the sending RTS normally issues an S-ACTIVITY-END.request. An activity end is an implicit major synchronization point and once successfully confirmed indicates to both RTSs that the RSDU has been secured. The sending RTS may then delete the transmitted RSDU.

If the receiving RTS cannot secure the RSDU, it must not confirm the activity end. It may either issue an S-U-EXCEPTION-REPORT.request or S-U-ABORT.request, depending upon the severity of its local problem (see clause 4.4.1 of the Standard).

F.3.4.1 S-ACTIVITY-END

Parameter	Req/Ind	Resp/Conf
Synchronization point Serial Number	1	
SS-User Data	–	–

1. The serial number of the implied major synchronization point is allocated by the Session-provider and passed up to both RTSs.

F.3.5 Other Actions of the Sending RTS during RSDU Transfer

If it detects a local system problem or receives an S-U-EXCEPTION-REPORT.indication (see F.4.1), the sending RTS may take one of the following actions:

I) Interrupt the current activity by issuing an S-ACTIVITY-INTERRUPT.request.

II) Discard the current activity by issuing an S-ACTIVITY-DISCARD.request.

III) A discarded activity cannot be resumed, and an RTS that receives such an indication should delete all knowledge and contents of the associated RSDU so far received. If the sending RTS wishes to re-transmit a previously discarded activity (RSDU), it must re-introduce it as a new activity (using S-ACTIVITY-START.request).

IV) Abort the session connection by issuing a S-U-ABORT.request, which also has the effect of interrupting the current activity.

F.4 Data Transfer Phase: Actions of Receiving RTS

F.4.1 Responding to Problems

If the receiving RTS detects a problem during receipt of an RSDU (for example, within an activity), it may issue either an S-U-ABORT.request or S-U-EXCEPTION-REPORT.request. Aborting the session connection is the most severe action; this is decribed in Section F.6.2. By issuing an S-U-EXCEPTION-REPORT.request, the receiving RTS indicates a less severe problem which it anticipates the sending RTS can overcome, and allows the sending RTS to take the most appropriate course of action. The actions the sending RTS may take are described in F.3.5.

In exceptional circumstances, the receiving RTS may have to delete a partially received activity, even though some minor synchronization points have been confirmed. In this case, the RTS responds to an S-ACTIVITY-START.indication for that activity by issuing an S-U-EXCEPTION-REPORT.request with the unrecoverable procedure error reason code. Alternatively, the receiving RTS may issue an S-U-ABORT.request.

F.4.1.1 S-U-EXCEPTION-REPORT

Parameter	**Req/Ind**
Reason	1
SS-User Data	–

1. This parameter may specify one of the following reasons:

   - receiving ability jeopardized,
   - local SS-User error,
   - sequence error,
   - unrecoverable procedure error,
   - non specific error.

F.4.2 Requesting Control of the Session Connection

When the RTS-user issues a RT-TURN-PLEASE.request, the RTS issues an S-TOKEN-PLEASE.request. This may be done either inside or outside an activity. Upon receipt of the S-TOKEN-PLEASE.Indication, the sending RTS issues a RT-TURN-PLEASE.indication to the RTS-user.

F.4.2.1 S-TOKEN-PLEASE

Parameter	**Req/Ind**
Tokens	1
SS-User Data	2

1. The receiving RTS will only request the data token. Since, for message handling, the tokens cannot be separated, the sending RTS always surrenders all of the other available tokens when issuing the S-CONTROL-GIVE.request.

2. This is the priority parameter of the RT-TURN-PLEASE.request primitive, defined as follows:

   Priority :: = INTEGER

To ensure that an RTS user's request for the turn is not lost during a Session abort/reconnection, the following rule applies. An RTS that has issued an S-TOKEN-PLEASE.request but not received an S-CONTROL-GIVE.indication should re-issue the S-TOKEN-PLEASE.request after session recovery.

F.5 Possible Actions of Sending RTS Outside an Activity

The sending RTS may choose to abort or release the session connection when there is no current activity, as described in Section F.6. It may also choose to send a new RSDU (or a previously discarded RSDU) by starting another activity and transferring the RSDU as described in F.3.1. Other actions are described in the following subsections.

F.5.1 Resuming an Interrupted Activity

The sending RTS may continue an RSDU that was previously interrupted by an S-ACTIVITY-INTERRUPT.request or user- or provider-initiated session abort. To do so, it uses the S-ACTIVITY-RESUME.request primitive. Parameters are supplied to link the continued activity with the previously interrupted part.

It is possible for the sending RTS to try to resume an activity that the receiving RTS believes to be complete. This happens when the session connection is aborted after the receiving RTS has confirmed the activity end, and the confirmation is lost. To overcome this, the following rules should be followed:

1. As the first activity started on a new connection, the sending RTS must attempt to resume the last interrupted activity (for example, the one that was current when the connection was aborted).

2. The receiving RTS should always record the Connection Id and Activity Id of the last RSDU which is completely secured.

3. The receiving RTS should check the details of each resumed activity to see if it coincides with the last RSDU the RTS secured. If it does, the receiving RTS should respond correctly to the sending RTS but discard the data it receives.

If the time to transfer an RSDU has expired, the sending RTS should generate an RT-EXCEPTION.Indication unless an S-ACTIVITY-END.confirm is outstanding. Since the RSDU might be completely received by the other RTS, the sending RTS should retain the RSDU until the activity is successfully resumed or discarded. This avoids the possibility of generating duplicate RSDUs.

F.5.1.1 S-ACTIVITY-RESUME

Parameter	**Req/Ind**
Activity Identifier	1
Old Activity Identifier	2
Synchronization Point Serial Number	3
Old Session Connection Identifier	4
SS-User Data	–

1. The issuing RTS must allocate and supply the next Activity Identifier number for the current session.

2. The RTS must supply the Activity Identifier assigned to the previously interrupted part of the RSDU.

3. The RTS will specify the Serial Number of the last confirmed checkpoint in the interrupted RSDU. The Session-provider will also set the current session serial number to this value. If there was no previously confirmed checkpoint, the activity (RSDU) cannot be resumed and thus must be discarded.

4. If the interrupted RSDU was suspended during a previous session connection (or on another current session connection), the Session Connection Identifier of that session must be supplied for recovery. The Session Connection Identifier of the previous (or current other) session connection is conveyed in the Calling SS-User Reference, Common Reference, and, optionally, Additional Reference Information Components of this parameter. The called SS-User Reference component is not used.

F.5.2 Giving Control of the Session to the Receiving RTS

When an RTS-user issues a RT-TURN-GIVE.request, the sending RTS will give control of the session to the other RTS using the S-CONTROL-GIVE.request primitive.

F.5.2.1 S-CONTROL-GIVE

The data, minor synch, and major/activity tokens are automatically passed to the other RTS.

Note F.3
This enables T.62 CSCC/RSCCP to be exchanged without parameters, that is, compatibly with CCITT Rec. T.62.

F.6 Session Connection Termination Phase

F.6.1 Orderly Release

The sending RTS issues an S-RELEASE.request in response to a RT-CLOSE.request. Upon receiving an S-RELEASE.indication, the RTS issues a RT-CLOSE.indication and an S-RELEASE.response.

Note F.4
The transport connection need not be released with the session connection and may be re-used.

F.6.1.1 S-RELEASE

Parameter	Req/Ind	Resp/Conf
Result	1	
SS-User Data	–	–

1. The responding RTS will always accept the release.

F.6.2 User Initiated Abort

If an RTS detects a serious problem, either inside or outside an activity, it may issue an S-U-ABORT.request. It should provide a reason parameter for diagnostic purposes, and if the abort was provoked by receipt of an invalid message handling parameter, the invalid parameter should also be reflected.

F.6.2.1 S-U-ABORT

Parameter	Req/Ind
User Data	1

1. The User Data parameter contains an AbortInformation data element, defined as follows:

```
AbortInformation :: = SET {
   [0] IMPLICIT AbortReason OPTIONAL,
   reflectedParameter [1] IMPLICIT BIT STRING OPTIONAL}

AbortReason :: = INTEGER {
   localSystemProblem(0),
   invalidParameter(1), -- reflectedParameter supplied
   unrecognizedActivity(2),
   temporaryProblem(3), -- the RTS cannot accept a session for a period of time --
   protocolError (4) -- RTS level protocol error --}
```

Note F.5
It is assumed that the session is aborted by the session provider upon detection of any unrecoverable error, that is, that S-P-EXCEPTION-REPORT.indication is not issued.

F.6.3 Provider Initiated Abort

The Session-provider may abort a session connection for any of a variety of reasons (for example, transport connection failure or local or remote provider problem), indicated by the reason parameter.

F.6.3.1 S-P-ABORT

Parameter	Ind
Reason	1

1. The following reason codes may be supplied:

- Transport disconnect,
- Protocol error,
- Undefined.

F.7 State Diagrams

F.7.1 Sending RTS

Figure F.1 specifies the legal interactions and relationships between the Session Service primitives when manipulated by the sending RTS.

F.7.2 Receiving RTS

Figure F.2 specifies the legal interactions and relationships between the Session service primitives when manipulated by the receiving RTS.

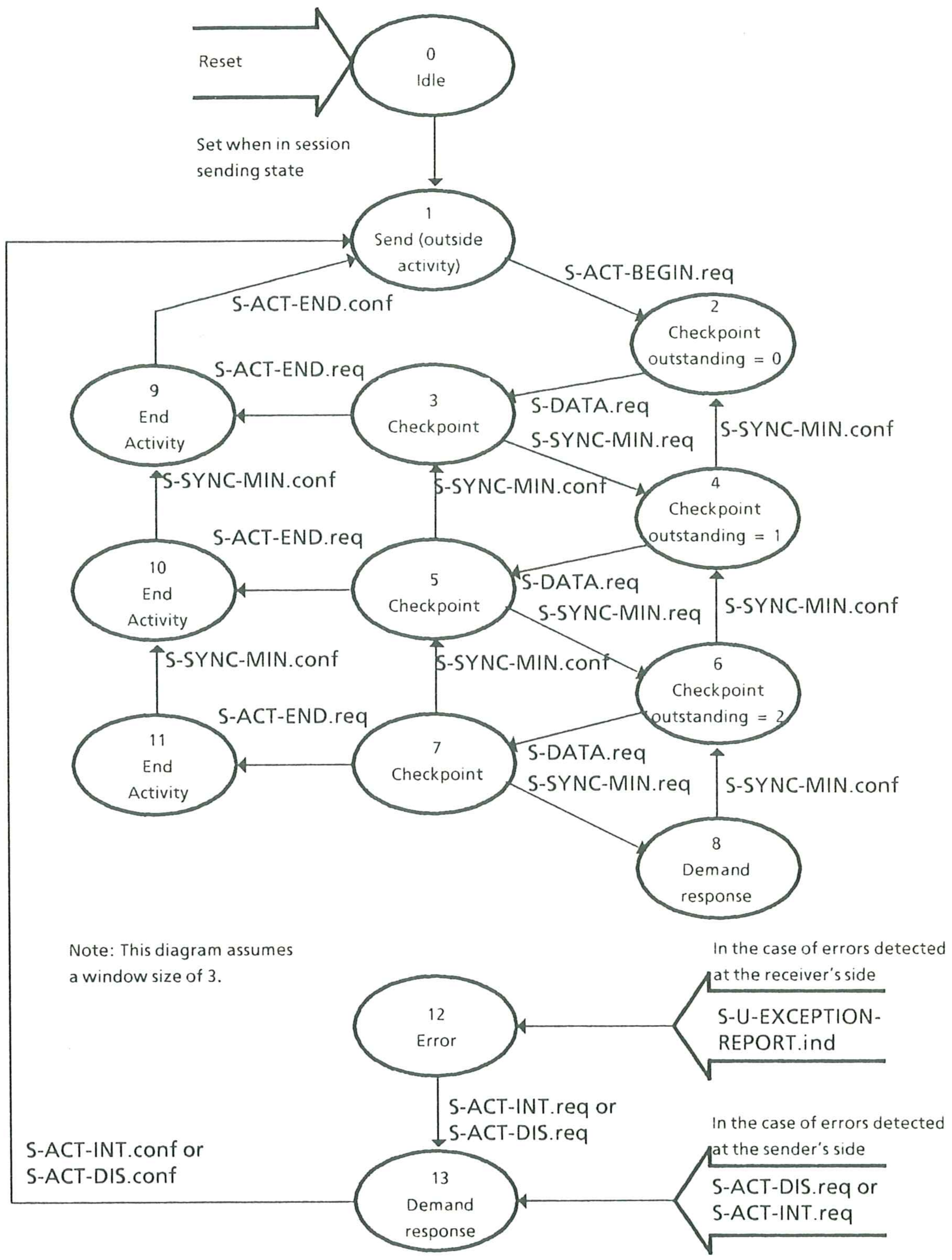


Figure F.1. State Diagram for allowed sequences of session service primitives for Sending RTS

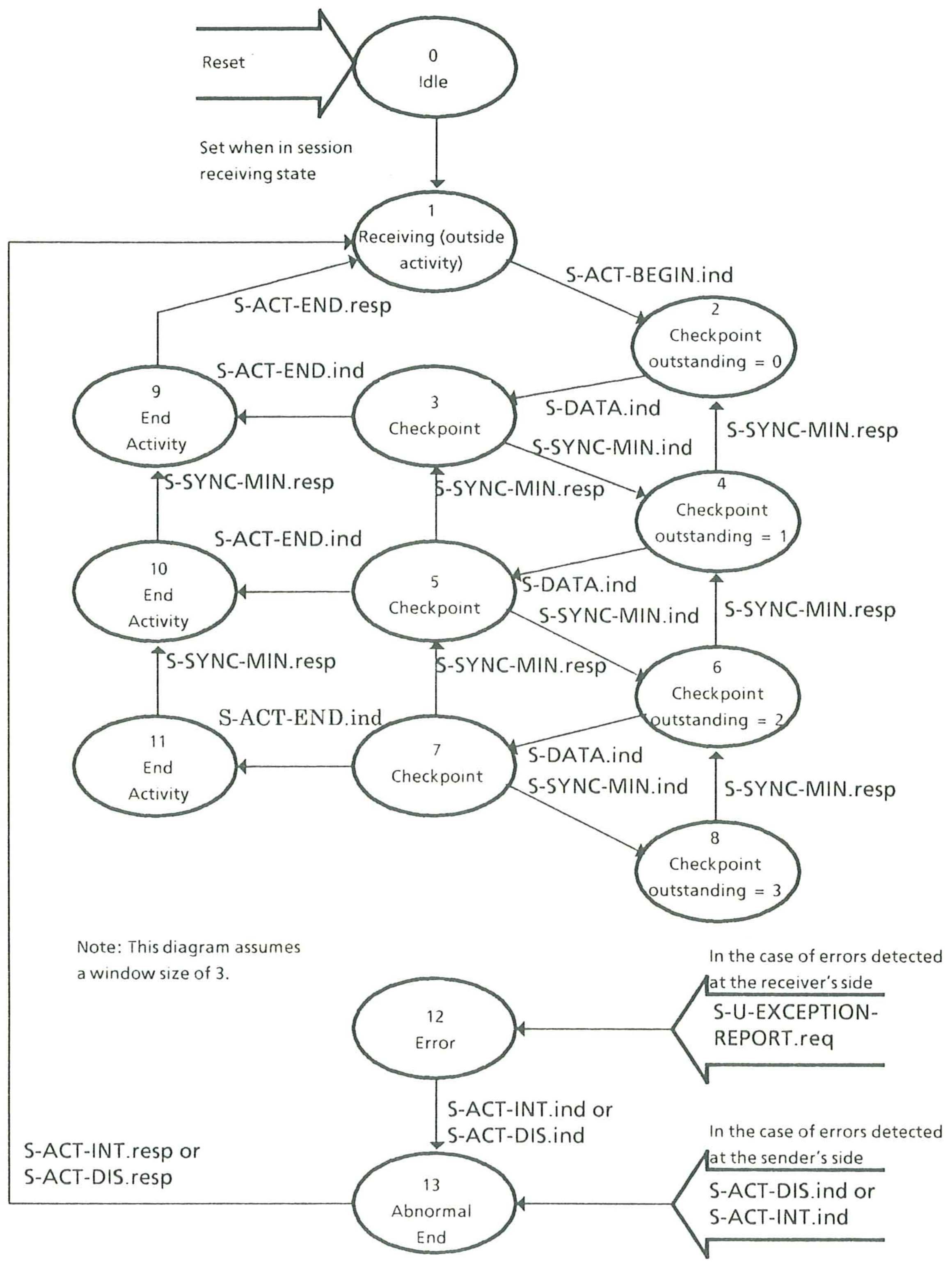

Reset
0
Idle
Set when in session receiving state
1
Receiving (outside activity)
S-ACT-BEGIN.ind
2
Checkpoint outstanding = 0
S-ACT-END.resp
S-ACT-END.ind
9
End Activity
3
Checkpoint
S-DATA.ind
S-SYNC-MIN.ind
S-SYNC-MIN.resp
4
Checkpoint outstanding = 1
S-SYNC-MIN.resp
S-SYNC-MIN.resp
S-ACT-END.ind
10
End Activity
5
Checkpoint
S-DATA.ind
S-SYNC-MIN.ind
S-SYNC-MIN.resp
S-SYNC-MIN.resp
S-SYNC-MIN.resp
6
Checkpoint outstanding = 2
S-ACT-END.ind
11
End Activity
7
Checkpoint
S-DATA.ind
S-SYNC-MIN.ind
S-SYNC-MIN.resp
8
Checkpoint outstanding = 3
Note: This diagram assumes a window size of 3.
In the case of errors detected at the receiver's side
S-U-EXCEPTION-REPORT.req
12
Error
S-ACT-INT.ind or S-ACT-DIS.ind
In the case of errors detected at the sender's side
S-ACT-DIS.ind or S-ACT-INT.ind
S-ACT-INT.resp or S-ACT-DIS.resp
13
Abnormal End

Figure F.2. State Diagram for allowed sequences of session service primitives for Receiving RTS

APPENDIX G

REALIZATION OF RTS BASED ON PRESENTATION AND BSS SESSION SERVICES

G.1 Introduction and Scope

This Appendix, which is part of the Standard, describes an implementation of the service specified in clause 3.3 of the Standard using standard ECMA-84.

The Service Subset DP-A is used by this implementation. The presentation layer entities use synchronized session services of the Session Layer protocol standard, ISO 8326. The terminology of these standards is used in this Appendix.

This realization transfers data structures and semantics unchanged between users of the RTS using Presentation services.

This implementation of the RTS specifies a specific usage of presentation services to transfer small or large volumes of data in a reliable manner.

G.2 Management of Presentation Connections

The establishment of an Association (RT-OPEN) between two RTS users involves the creation of a presentation connection (ps-connection) between the two related RTS entities. The end of an Association (RT-CLOSE) involves the termination of the related ps-connection.

Between the beginning and the end of an Association an abnormal termination of the ps-connection may occur. In that case it is the responsibility of the RTS entities to re-establish the ps-connection (see Recovery). At any moment there is at most one ps-connection in the association.

1. Ps-connections involved in an Association are two-way alternate.

2. Initial turn is with the initiator of the Association called Initiator. The acceptor is called Responder.

3. The condition for refusal of a ps-connection is outside the scope of this specification. They are a local matter of each RTS entity.

This Standard allows the "idling" of a ps-connection (i.e. ps-connections may be kept while the RTS entity has no RSDU to transfer).

G.3 Data Transfer Phase

The data transfer phase is initiated on the occurrence of an RT-TRANSFER.request. RDSU's are transferred as transparent data of the P-DATA service. In case of congestion, a receiving RTS may use back pressure flow control to limit the rate at which it receives RSDU's.

To request and confirm the correct safe storage of RSDU's , the session synchronization services which are transparently made available as part of the presentation services are used. For this, each RSDU may (but need not) be fragmented into a number of PSDU's. The maximum size of a fragment is negotiated at the ps-connection establishment time and is called "checkpoint size".The default value is infinite.

To each PSDU which is the last fragment of an RSDU a major synchronization point is attached. To each PSDU which is not the last fragment of an RSDU a minor (normal) synchronization point is attached.

The confirmation of a synchronization point indicates that the corresponding PSDU has been safely stored by the receiving RTS entity. The transfer of a RSDU is considered complete when the major synchronization point has been acknowledged.

The maximum number of unconfirmed minor synchronization points is negotiated at the ps-connection establishment time and is called window size. The sending RTS may issue P-DATA requests and P-SYNC requests unless the agreed window size has been reached.

The confirmation of a synchronization point does not indicate that any processing has been done. The procedure applying when subsequent processing reveals a problem in the RSDU are outside the scope of the RTS. (In the MIDA context, an SMPDU will be created and sent over another association).

Optionally the RTS offers its users a TWA mode of operation. In this case the right to transfer a PSDU is mediated by the RTS token. The RTS entities use the (transparently available) session token services to realize this facility; all session tokens available are moved simultaneously. The token is not moved if there is an incompletely transferred RSDU. The RTS entity owing the RTS token is called Sender, an RTS entity not owing the RTS token is called Receiver.

G.4 Recovery

By preference this implementation tries to transfer one RSDU in one dialogue unit without re-synchronization.

If this is not possible due to internal reasons for one of the RTS entities, re-synchronization may be requested (P-RESYNC), implying retransmission from the agreed synchronization point. This is preferably the synchronization point which has most recently been confirmed by the Receiver. If the re-synchronization point coincides with the beginning of the dialogue unit it is not guaranteed that the same RSDU is transferred.

If the transmission of an RSDU is interrupted by a P-DISCONNECT or P-ABORT, the Initiator and the Responder should agree on the allocation of the roles of Sender and Receiver (if applicable) and on the synchronization points from which recovery will start using the following rules:

1. Establishment of the new connection is initiated by the Initiator, the first dialogue unit of the new connection is the continuation of the interrupted dialogue unit.

2. The Initiator requests the recovery to continue as follows:

   a) The Initiator identifies the ps-connection to be continued (this is the most recent P-CONNECT.request for which a P-CONNECT.confirmation was received).

   b) If the Initiator is the Sender, or if the service operates in the monologue-mode, continuation is requested to be from the most recently requested synchronization point (or any preceeding synchronization point up to and including the most recently requested major synchronization point).
   In this case the Initiator keeps the token.

   c) If the Initiator is Receiver, the continuation is requested to be from the most recent synchronization point responded to by the Initiator or any preceeding synchronization point up to and including the most recent resonded major synchronization point.

   In this case the Initiator:

   - gives the token to the Responder if the Initiator has issued an P-TOKENS-GIVE with no confirmation that the tokens were received and

- leaves the choice to the Responder if such confirmation was obtained. (Receipt of data serves as confirmation that the tokens were received).

3. The Responder determines the recovery to continue as follows:

   a) The Responder confirms the identification of the ps-connection to be continued.

   b) If the Initiator keeps the token (2 b above), the Responder selects continuation to be from the most recent synchronization point responded to by the Responder, with a number lower than or equal to the synchronization point identified in the CONNECT.indication or any preceeding synchronization point up to and including the most recently responded major synchronization point.

   c) If the Initiator does not keep the token (2 c above), the Responder selects continuation from the synchronization point proposed by the Initiator.
   If the choice for the allocation of the token is to the Responder, it keeps the tokens only if it was the Sender;
   otherwise the Responder indicates that the Initiator becomes the Sender.

G.5 Normal Termination of an Association

Termination of an Association is initiated by the RTS user by means of an RT-CLOSE service primitive. This is carried out by the presentation release service. Only the Sender RTS may request termination. Oherwise the RT-CLOSE is rejected (local event).

The Initiator RTS may request this even though several synchronization points are at this time unconfirmed. RSDUs submitted to the RTS service for transmission prior to the RT-CLOSE will be transmitted before the association is closed.

G.6 Completion of Reliable Transfer

The reliable transfer of an RDSU is considered by the Receiver to be complete if and only if it has received an P-END-DU.indication. Only in that situation the RTS-provider will issue an RT-TRANSFER.indication and pass the RSDU to the RTS-user.

The reliable transfer of an RSDU is considered by the Sender to be complete, if and only if it has received an P-END-DU.confirmation.

As long as the Sender has no proof that the reliable transfer of the RSDU has been completed and signalled to the addressed RTS-user, it should take the initiative to continue the transfer, unless the ps-connection is terminated, in which case the Initiator is responsible to establish the new connection. Note that under exceptional conditions a message may be delivered twice from the RTS to the RTS-user.

G.7 RTS Time Out

If the time-out for an RSDU expires before the P-END-DU.request for the RSDU is generated, the Sender will take actions to destroy all information related to the RSDU in the Receiver. This is carried out by means of P-RESYNC services returning to the most recent major synchronization point. On receipt of a P-RESYNC indication the Responder should destroy the received part of the RSDU. The Initiator will also (immediately) generate an RT-EXCEPTION .indication with Diagnostic = time-out.

G.8 Presentation Services Used and Summary

In summary, the Initiator RTS entity takes the initiatives necessary to transfer a RSDU submitted to it by its user :

- it creates ps-connections as and when deemed necessary;

- it initiates the subsequent ps-services necessary to realize the transfer.

This is carried out by the following services. These services are defined in ECMA-84:

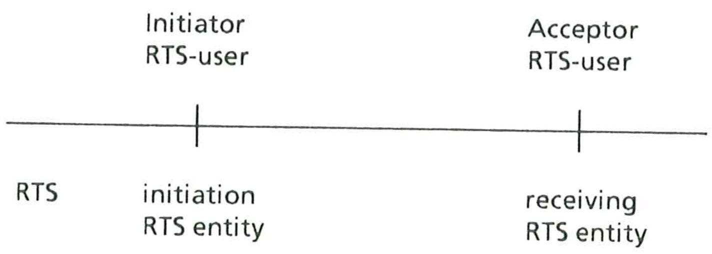
P-CONNECT
P-RELEASE
P-DISCONNECT
P-ABORT
P-DATA
P-SYNC
P-END-DU
P-RESYNC
P-PLEASE
P-TOKENS-GIVE

G.8.1 P-CONNECT

An RTS entity attempts presentation connection establishment in response to either of the following :

1. OPEN	An RT-OPEN.request issued by the RTS-USER.
2. RECOVER	An internal rule when attempting to recover a ps-connection that was aborted.

The two cases are distingueshed by means of data elements in the transparent-data parameter.



Parameter	**Req/Ind**	**Resp/Conf**
Service ident	0	
Service version	0	0
Service subset	0	0
Connection identifier	1	1
Compression	0	0
Special convention	0	0
Conventions user caller	2	
Convention user called	2	2
Diagnostic		3
Quality of service	4	
Session requirements	5	5
Initial synchronization point serial number	6	6
Initial assignment of tokens	7	7
Transparent-data	8	8

0. Service ident : DPS
   Service version : 1
   Service subset : DP-A
   Compression : OFF
   Special conventions : not used

1. The initiating RTS will supply a session connection identifier, which will be used to uniquely identify the connection. This parameter consists of :

```
Session Connection identifier :: = SEQUENCE {
        initiatingRTSaddress [0] IMPLICIT PrintableString
        [1] IMPLICIT UTCTime,
        additionalInformation [2] IMPLICIT PrintableString OPTIONAL }
```

2. Calling PSAP is the address of the initiating RTS entity. It is equivalent to the RT-OPEN initiator address parameter.
   Called PSAP is the address of the receiving RTS entity. It is equivalent to the RT-OPEN responder address parameter.

3. The receiving RTS may choose to accept or reject the P-CONNECT.indication. The result parameter indicates which has occured. Rejections by a called SS-User distinguish different reasons for failure :

   - Reason not specified.
   - Rejections due to temporary congestion.
   - Other reason : the user data may be used to provide further information.

4. Quality of service.

   The parameters Extended Control and Optimized DialogueTransfer are set to required. The first permits the use of expedited flow in the case congestion for ABORT and Re-synchronizE. The second parameter allows concatenation of service commands in the session protocol, especially Data transfer and Minor synchronization.

5. Session requirements.

   This parameter specifies the functional units to be used :

   - half duplex f.u. (indicates the existence of data token),
   - minor synchronization f.u.
   - major synchronization f.u.
   - re-synchronize f.u.

6. Initial synchronization point serial number is set to zero in the OPEN case. It is set as described in G.4 for the RECOVER case.

7. Initial assignment of tokens.

   In the OPEN-case the Data token, Sync minor token and Major token are set as indicated in the initial turn parameter of the RT-OPEN.request. This may only identify the initiator (i.e. the Initiator) in this implementation.
   In the RECOVER-case the tokens are set as described in G.4.

8. Transparent-data

   In the P-CONNECT.request, this parameter contains a P-CONNECT data element. In the P-CONNECT.response, it contains a PAccept or PRefuse data element depending upon the diagnostic parameter.

These data elements are defined as follows:

```
PConnect :: = SET {
        [0]  IMPLICIT DataTransferSyntax
        [1]  IMPLICIT pUserData  SET {
             checkpointSize [0] IMPLICIT INTEGER DEFAULT 0,
                  -- zero means no checkpointing
             windowSize [1]  IMPLICIT INTEGER DEFAULT 3,
             dialogueMode [2] IMPLICIT INTEGER
                  { monologue (0), twa(1) DEFAULT monologue},
             [3] connectionData,
             applicationProtocol [4] IMPLICIT INTEGER {p1(1), MIDAP1(2)}}}
                  -- It comes from the RT-OPEN.request-parameter

  PAccept :: = SET {
        [0]  IMPLICIT DatatransferSyntax,
        [1]  IMPLICIT pUserData  SET {
             checkpointSize [0] IMPLICIT INTEGER DEFAULT 0,
             windowSize [1]  IMPLICIT INTEGER DEFAULT 3,
             [2] connectionData }}

  PRefuse :: =  SET { [0] IMPLICIT RefuseReason}

  DatatransferSyntax :: =  SET {[0] IMPLICIT INTEGER {mhs4 (0)}}

  ConnectionData :: =  CHOICE {
       Open [0] ANY,         -- that is, RTS user data SEQUENCE
       Recover [1]  IMPLICIT Session Connection Identifier}

  Refuse Reason :: =  INTEGER {
       rtsBusy (0),
       cannotRecover (1),
       validationFailure (2),
       unacceptableDialogueMode (3),
       unacceptableProtocol (4)}
```

G.8.2 Data transfer phase

A dialogue unit is structured as shown in Figure G.1.

G.8.2.1 P-DATA

The normal data transfer service allows the Sender RTS to transfer PSDU.

Parameter	Req/Ind
Formatted data	1
Transparent data	2

1. Not used
2. It contains one fragment of the RSDU

G.8.2.2 P-SYNC

These service primitives are the means for sending RTS entities to define a minor synchronization point and for the receiving RTS to confirm this synchronization.

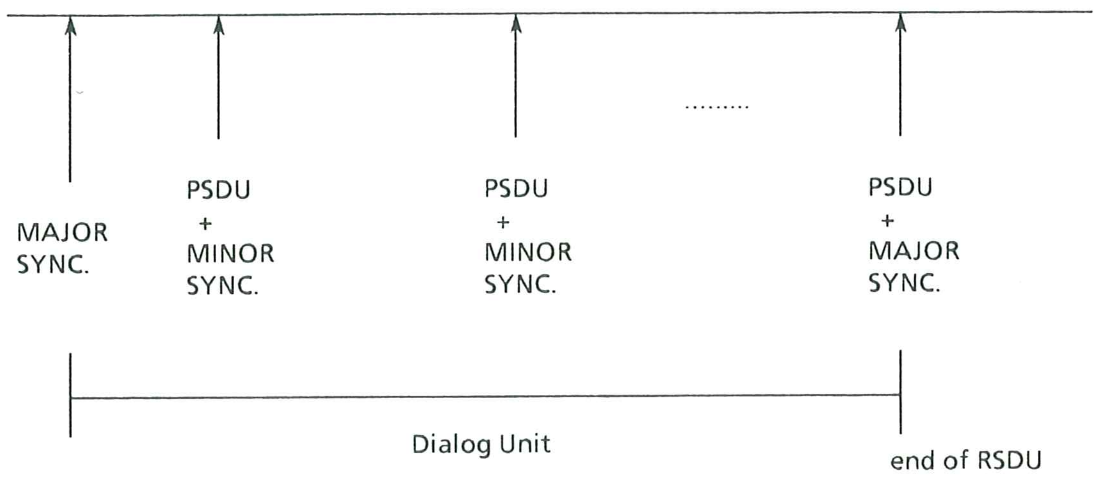


Figure G.1. Structure of a dialogue unit

Parameter	**Req/Ind**	**Resp/Conf**
Type	1	
Synchronization point serial number	2	2
SS-User data	3	3

1. The RTS uses only the explicit confirmation expected type of minor synchronization. Each minor synchronization must be confirmed in the orders received by the receiving RTS. If the receiving RTS has detected a problem, it should not confirm the minor synchronization, but rather should issue an S-RESYNCHRONIZE or an S-U-ABORT depending upon the severity of the problem.

2. The session service provider allocates checkpoint serial numbers and passes them to the sending and receiving RTS to associate with the transmitted data.

3. Not used.

G.8.2.3 P-END-DU

These service primitives are the means for a sending RTS entity to indicate the end of the transfer of an RSDU to the other RTS entitiy.

Parameter	**Req/Ind**	**Resp/Conf**
Synchronization point serial number	1	
SS-User data	2	2

1. The session service provider allocates serial numbers and passes them to the sending and receiving RTS to associate with the transmitted data.

2. Not used.

G.8.2.4 Re-synchronization : P-RESYNC

These service primitives are the means for both sending and receiving RTS entities to request the retransmission of the RSDU in transit from one of the synchronization points.

The synchronization point in the response and confirmation identifies the actual synchronization point from which retransmission will continue.

Parameter	Req/Ind	Resp/Conf
Re-synchronize type	1	
Synchronization point serial number	2	2
Tokens	3	3
SS-User data	4	4

1. Restart

2. Set to appropriate value (any minor synchronization point after the last major confirmed synchronization point or that latest one)

3. Set so that the sending RTS keeps the tokens.

4. Not used.

G.8.2.5 Requesting the tokens of the Presentation Connection: P-PLEASE

When the RTS-user issues an RT-TURN-PLEASE.request, the RTS issues a P-PLEASE.request. Upon receipt of the P-PLEASE.indication, the sending RTS issues an RT-TURN-PLEASE.indication to the RTS-user.

Parameter	Req/Ind
Tokens	1
SS-User data	2

1. The receiving RTS will request all available tokens.

2. This is the priority parameter of the RT-TURN-PLEASE.request primitive, defined as follows:

   priority :: = INTEGER

   To ensure that an RTS users request for the turn is not lost during a Presentation abort/re-connection, the following rule applies:

   An RTS that has issued a P-PLEASE.request, but not received a P-TOKENS-GIVE.indication should re-issue the P-PLEASE.request after recovery of the ps-connection.

G.8.2.6 Transferring the tokens of the Presentation Connection: P-TOKENS-GIVE

When an RTS-user issues an RT-TURN-GIVE.request, the sending RTS will give the tokens of the presentation connection to the other RTS using the P-TOKENS-GIVE.request primitive. Upon receipt of the P-TOKENS-GIVE.indication the RTS issues an RT-TURN-PLEASE.indication to the RTS-user.

Parameter	Req/Ind
Tokens	1

1. The sending RTS gives all available tokens when issuing a P-TOKENS-GIVE.request.

G.8.3 Presentation Connection Termination Phase

G.8.3.1 Orderly Release : P-RELEASE

These service primitives are used by the Sender to remove a connection in an orderly manner, as and when deemed necessary.

Parameter	Req/Ind	Resp/Conf
Diagnostic	1	
Transparent data	2	2

1. The parameter result is set to affirmative (the release token is not defined).

2. Not used.

G.8.3.2 User-initiated-abort : P-DISCONNECT

This service primitive is used by either the Initiator or the Responder to obtain an unclean termination of the connection whenever that is deemed needed.

Parameter	Req/Ind
Transparent data	1

1. Not used.

G.8.3.3 Provider Initiated Abort : P-ABORT

This service primitive is used by the service provider to indicate an unclean termination of the connection.

Parameter	Indication
Diagnostic	1

1. The following reason codes may be supplied :

   - Transport disconnect,
   - Protocol error,
   - Undefined.

G.9 Usage of Session Service Primitives

Table G.1 gives the mapping between Presentation Services and Session Services.

Presentation	Session
P-CONNECT	S-CONNECT
P-RELEASE	S-RELEASE
P-DISCONNECT	S-U-ABORT
P-ABORT	S-P-ABORT
P-DATA	S-DATA
P-SYNC	S-SYNC-MINOR
P-END-DU	S-SYNC-MAJOR
P-RESYNC	S-RESYNCRONIZE
P-PLEASE	S-PLEASE
P-TOKENS-GIVE	S-TOKENS-GIVE

Table G.1. Mapping between Presentation and Session Services