X/Open CAE Specification

API to Electronic Mail (X.400)

Issue 2

X/Open Company Ltd.

X/Open CAE Specification

API to Electronic Mail (X.400) Issue 2

ISBN: ISBN 1-85912-009-1

X/Open Document Number: C316

Published by X/Open Company Ltd., U.K.

Any comments relating to the material contained in this document may be submitted to X/Open at:

X/Open Company Limited
Apex Plaza
Forbury Road
Reading
Berkshire, RG1 1AX
United Kingdom

or by Electronic Mail to:

XoSpecs@xopen.co.uk

Contents

List of Figures

List of Tables

Preface

X/Open

X/Open is an independent, worldwide, open systems organisation supported by most of the world's largest information systems suppliers, user organisations and software companies. Its mission is to bring to users greater value from computing, through the practical implementation of open systems.

X/Open's strategy for achieving this goal is to combine existing and emerging standards into a comprehensive, integrated, high-value and usable open system environment, called the Common Applications Environment (CAE). This environment covers the standards, above the hardware level, that are needed to support open systems. It provides for portability and interoperability of applications, and so protects investment in existing software while enabling additions and enhancements. It also allows users to move between systems with a minimum of retraining.

X/Open defines this CAE in a set of specifications which include an evolving portfolio of application programming interfaces (APIs) which significantly enhance portability of application programs at the source code level, along with definitions of and references to protocols and protocol profiles which significantly enhance the interoperability of applications and systems.

The X/Open CAE is implemented in real products and recognised by a distinctive trade mark — the X/Open brand — that is licensed by X/Open and may be used on products which have demonstrated their conformance.

X/Open Technical Publications

X/Open publishes a wide range of technical literature, the main part of which is focussed on specification development, but which also includes Guides, Snapshots, Technical Studies, Branding/Testing documents, industry surveys, and business titles.

There are two types of X/Open specification:

- *CAE Specifications*

  CAE (Common Applications Environment) specifications are the stable specifications that form the basis for X/Open-branded products. These specifications are intended to be used widely within the industry for product development and procurement purposes.

  Developers who base their products on a current CAE specification can be sure that either the current specification or an upwards-compatible version of it will be referenced by a future X/Open brand (if not referenced already), and that a variety of compatible, X/Open-branded systems capable of hosting their products will be available, either immediately or in the near future.

  CAE specifications are published as soon as they are developed, not published to coincide with the launch of a particular X/Open brand. By making its specifications available in this way, X/Open makes it possible for conformant products to be developed as soon as is practicable, so enhancing the value of the X/Open brand as a procurement aid to users.

- *Preliminary Specifications*

  These specifications, which often address an emerging area of technology and consequently are not yet supported by multiple sources of stable conformant implementations, are released in a controlled manner for the purpose of validation through implementation of products. A Preliminary specification is not a draft specification. In fact, it is as stable as X/Open can make it, and on publication has gone through the same rigorous X/Open development and review procedures as a CAE specification.

  Preliminary specifications are analogous to the *trial-use* standards issued by formal standards organisations, and product development teams are encouraged to develop products on the basis of them. However, because of the nature of the technology that a Preliminary specification is addressing, it may be untried in multiple independent implementations, and may therefore change before being published as a CAE specification. There is always the intent to progress to a corresponding CAE specification, but the ability to do so depends on consensus among X/Open members. In all cases, any resulting CAE specification is made as upwards-compatible as possible. However, complete upwards-compatibility from the Preliminary to the CAE specification cannot be guaranteed.

In addition, X/Open publishes:

- *Guides*

  These provide information that X/Open believes is useful in the evaluation, procurement, development or management of open systems, particularly those that are X/Open-compliant. X/Open Guides are advisory, not normative, and should not be referenced for purposes of specifying or claiming X/Open conformance.

- *Technical Studies*

  X/Open Technical Studies present results of analyses performed by X/Open on subjects of interest in areas relevant to X/Open's Technical Programme. They are intended to communicate the findings to the outside world and, where appropriate, stimulate discussion and actions by other bodies and the industry in general.

- *Snapshots*

  These provide a mechanism for X/Open to disseminate information on its current direction and thinking, in advance of possible development of a Specification, Guide or Technical Study. The intention is to stimulate industry debate and prototyping, and solicit feedback. A Snapshot represents the interim results of an X/Open technical activity. Although at the time of its publication, there may be an intention to progress the activity towards publication of a Specification, Guide or Technical Study, X/Open is a consensus organisation, and makes no commitment regarding future development and further publication. Similarly, a Snapshot does not represent any commitment by X/Open members to develop any specific products.

Versions and Issues of Specifications

As with all *live* documents, CAE Specifications require revision, in this case as the subject technology develops and to align with emerging associated international standards. X/Open makes a distinction between revised specifications which are fully backward compatible and those which are not:

- a new *Version* indicates that this publication includes all the same (unchanged) definitive information from the previous publication of that title, but also includes extensions or additional information. As such, it *replaces* the previous publication.

- a new *Issue* does include changes to the definitive information contained in the previous publication of that title (and may also include extensions or additional information). As such, X/Open maintains *both* the previous and new issue as current publications.

Corrigenda

Most X/Open publications deal with technology at the leading edge of open systems development. Feedback from implementation experience gained from using these publications occasionally uncovers errors or inconsistencies. Significant errors or recommended solutions to reported problems are communicated by means of Corrigenda.

The reader of this document is advised to check periodically if any Corrigenda apply to this publication. This may be done either by email to the X/Open info-server or by checking the Corrigenda list in the latest X/Open Publications Price List.

To request Corrigenda information by email, send a message to info-server@xopen.co.uk with the following in the Subject line:

```
request corrigenda; topic index
```

This will return the index of publications for which Corrigenda exist.

This Document

This document is a CAE Specification (see above). It defines the application programming interfaces (APIs) to Electronic Mail (X.400). These APIs provide for access to, and interconnection of, messaging systems whose architecture is in accordance with the CCITT/ISO X.400 series of standards.

This *Issue 2* of the X.400 CAE Specification includes revisions to align with the IEEE X.400 API group of standards that themselves are based on the previous version of this X/Open specification.

All new implementation work by API providers should be based on this Issue 2. The previous specification will be retained by X/Open for only so long as there is demand to brand products based on it.

X.400 is one of several specifications that X/Open originally developed in collaboration with the X.400 API Association. The other documents are XOM, XDS, XMS and XEDI API specifications, and a Guide to Selected X.400 and Directory Services APIs.

The XOM and XDS specifications have similarly served as bases for corresponding IEEE standards. X/Open has now also revised the XOM and XDS specifications into *Issue 2* publications, to align them with the corresponding IEEE Standards.

Structure

This document is organised as follows:

- Chapter 1 introduces the interfaces and their specifications. It indicates the purposes of the interfaces, provides an overview of them, identifies the levels of abstraction at which the interfaces are defined, explains how identifiers at one level of abstraction are derived from those at the other, overviews the facilities provided by the OM API, introduces the concept of features, summarises the service implementation options, gives the conformance requirements imposed upon manufacturers and their products, and lists the abbreviations the document uses.

- Chapter 2 describes the overall functional architecture of the systems whose construction the MA and MT interfaces are intended to facilitate. the MA interface either makes the functionality of the MTS accessible to an MS or a UA, or makes the functionality of a simple MS accessible to a UA. The MT interface, on the other hand, divides an MTA into two software components.
- Chapter 3 defines the MA and MT interfaces. It specifies the functions that the service makes available to the client, the data types of which the arguments and results of the functions are data values, and the return codes that denote the outcomes (in particular, the exceptions) that the functions may report. The chapter also summarises the declarations that define the C interfaces.
- Chapter 4 defines the IM 84 Package and the IM 88 Package. The first provides the functionality of IM (1984), the second that of IM (1988).
- Chapter 5 defines the MH 84 Package and the MH 88 Package. The first provides the functionality of MH (1984), the second that of MH (1988).
- Chapter 6 defines the SM 88 Package, which provides the functionality of SM (1988).
- Appendix A describes how, in the context of selected operating systems, the C implementation of a client may be bound at run time to the C implementation of the service.
- Appendix B introduces X.400 and its terms and abbreviations, for the information of readers who may be unfamiliar with X.400.
- Appendix C identifies some of the issues associated with designing a mail system gateway thast is capable of making use of the MT interface that this document defines.

A glossary and index are provided.

Typographical Conventions

The following typographical conventions are used throughout this document:

- **Bold** font is used in text for options to commands, filenames, keywords, type names, data structures and their members, and language-independent names.
- *Italic* strings are used for emphasis or to identify the first instance of a word requiring definition. Italics in text also denote:
  - — command operands, command option-arguments or variable names, for example, substitutable argument prototypes
  - — environment variables, which are also shown in capitals
  - — utility names
  - — external variables, such as *errno*
  - — functions; these are shown as follows: *name*( ). Names without parentheses are C external variables, C function family names, utility names, command operands or command option-arguments.
- Roman font is used for the names of constants and literals.
- The notation **<file.h>** indicates a header file.
- Names surrounded by braces, for example, {ARG_MAX}, represent symbolic limits or configuration values which may be declared in appropriate headers by means of the C

#define construct.

- The notation [EABCD] is used to identify a return value ABCD, including if this is an an error value.
- Syntax, code examples and user input in interactive examples are shown in `fixed width` font. Brackets shown in this font, `[ ]`, are part of the syntax and do *not* indicate optional items.
- For a more detailed description of the C language binding font usage, see Chapter 2.

Trade Marks

X/Open™ and the ''X'' device are trade marks of X/Open Company Ltd.

Referenced Documents

This section identifies other documents upon which the present document depends.

The following document defines the OM interface:

- **XOM** - OSI-Abstract-Data Manipulation API, CAE Specification, X/Open Company Limited in conjunction with X.400 API Association, 1991.

The following documents constitute X.400 (1984). Developed under the auspices of, and ratified by, the International Telegraph and Telephone Consultative Committee (CCITT), they provide, together with ISO/DIS 8883 (see below), the functional basis for the MA and MT interfaces and the IM 84 and MH 84 Packages.

- Recommendation X.400, Message Handling Systems: System Model, Service Elements, CCITT Red Book, Fascicle VIII.7, International Telecommunications Union, October 1984, pp. 3-38.
- Recommendation X.401, Message Handling Systems: Basic Service Elements and Optional User Facilities, Ibid., pp. 39-45.
- Recommendation X.408, Message Handling Systems: Encoded Information Type Conversion Rules, Ibid., pp. 45-61.
- Recommendation X.409, Message Handling Systems: Presentation Transfer Syntax and Notation, Ibid., pp. 62-93.
- Recommendation X.410, Message Handling Systems: Remote Operations and Reliable Transfer Service, Ibid., pp. 93-126.
- Recommendation X.411, Message Handling Systems: Message Transfer Layer, Ibid., pp. 127-182.
- Recommendation X.420, Message Handling Systems: Interpersonal Messaging User Agent Layer, Ibid., pp. 182-219.
- Recommendation X.430, Message Handling Systems: Access Protocol for Teletex Terminals, Ibid., pp. 219-266.
- X.400-Series Implementors' Guide, Version 6, International Telecommunications Union, 6 November 1987.

The following two documents from the Message Oriented Text Interchange System (MOTIS), developed under the auspices of, but not fully ratified by, the International Organisation for Standardisation (ISO), provide (only) the basis for the definitions of the internal trace information and internal trace entries of the MH 84 Package.

- ISO/DIS 8883, Information Processing (Text Communication) Message-Oriented Text Interchange System, Message Transfer Sublayer, Message Interchange Service and Message Transfer Protocol, International Organisation for Standardisation, 1986.
- ISO/DIS 9065, Information Processing - Text Communication - Message Oriented Text Interchange System User Agent Sublayer - Inter-personal messaging user agent - Message interchange formats and protocols: December 1986.

The following documents constitute X.400 (1988) which significantly extends the functionality of X.400 (1984). Developed under the auspices of, and ratified by, the CCITT, these documents provide the terminology for the interfaces and packages. Together with the previous list of Recommendations they also provide the functional basis for the MA and MT interfaces and the IM 88, MH 88 and SM 88 Packages.

- Recommendation X.400, Message Handling: System and Service Overview, CCITT Blue Book, Fascicle VIII.7, International Telecommunications Union, 1988. (See also ISO 10021-1.)
- Recommendation X.402, Message Handling Systems: Overall Architecture, Ibid. (See also ISO 10021-1.)
- Recommendation X.403, Message Handling Systems: Conformance Testing, Ibid.
- Recommendation X.407, Message Handling Systems: Abstract Service Definition Conventions, Ibid. (See also ISO 10021-3.)
- Recommendation X.408, Message Handling Systems: Encoded Information Type Conversion Rules, Ibid. (See also ISO 10021-1.)
- Recommendation X.411, Message Handling Systems: Message Transfer System: Abstract Service Definition and Procedures, Ibid. (See also ISO 10021-4.)
- Recommendation X.413, Message Handling Systems: Message Store: Abstract Service definition, Ibid. (See also ISO 10021-5.)
- Recommendation X.419, Message Handling Systems: Protocol Specifications, Ibid. (See also ISO 10021-6.)
- Recommendation X.420, Message Handling Systems: Interpersonal Messaging System, Ibid. (See also ISO 10021-7.)
- MHS Implementor's Guide, Version 10; Source: CCITT Special Rapporteur Group on Message Handling Systems (Question 18/VII) and ISO/IEC JTC 1/SC 18/WG 4 SWG on Messaging, International Telecommunications Union, February 1993.

The following other documents supplement X.400. Developed under the auspices of, and ratified by, the CCITT, they provide character set (ITA2, IA5 and Teletex), encoded information type (Group 3 (G3) facsimile, G4 facsimile and Videotex), and addressing (X.121) definitions for the IM and MH packages.

- Recommendation F.1, Operational Provisions for the International Public Telegram Service, CCITT Red Book, Fascicle II.4, International Telecommunications Union, October 1984.
- Recommendation F.200, Teletex Service, Fascicle II.5, Ibid.
- Recommendation T.4, Standardisation of Group 3 Facsimile Apparatus for Document Transmission, Fascicle VII.3, Ibid.
- Recommendation T.5, General Aspects of Group 4 Facsimile Apparatus, Fascicle VII.3, Ibid.
- Recommendation T.6, Facsimile Coding Schemes and Coding Control Functions for Group 4 Facsimile Apparatus, Fascicle VII.3, Ibid.
- Recommendation T.30, Procedures for Document Facsimile Transmission in the General Switched Telephone Network, Fascicle VII.3, Ibid.
- Recommendation T.50, International Alphabet No.5, Fascicle VII.3, Ibid.
- Recommendation T.60, Terminal Equipment for Use in the Teletex Service, Fascicle VII.3, Ibid.

- Recommendation T.61, Character Repertoire and Coded Character Sets for the International Teletex Service, Fascicle VII.3, Ibid.
- Recommendation T.62, Control Procedures for Teletex and Group 4 Facsimile Services, Fascicle VII.3, Ibid.
- Recommendation T.100, International Information Exchange for Interactive Videotex, Fascicle VII.3, Ibid.
- Recommendation T.101, International Interworking for Videotex Services, Fascicle VII.3, Ibid.
- Recommendation T.400, Introduction to Document Application Transfer and Manipulation, Fascicle VII.6, Ibid.
- Recommendation T.501, A Document Application Profile MM for the Interchange of Formatted Mixed-mode Documents, Fascicle VII.7, Ibid.
- Recommendation T.503, A Document Application Profile for the Interchange of Group 4 Facsimile Documents, Fascicle VII.7, Ibid.
- Recommendation X.121, International Numbering Plan for Public Data Networks, Fascicle VIII.4, Ibid.

The following document also supplements X.400. Developed under the auspices of, and ratified by, ISO, it provides additional addressing definitions for the MH packages.

- ISO 3166, Codes for the Representation of Names of Countries, International Organisation for Standardisation.

The following document defines the X.500 Directory API. It defines the Certificates, Name and Presentation Address classes upon which the IM, MH and SM packages depend.

- **XDS** - API to Directory Services, CAE Specification, X/Open Company Limited in conjunction with X.400 API Association, 1991.

The following document defines the Portable Operating System Interface for Computer Environments (POSIX), standardised by the Institute of Electrical and Electronics Engineers (IEEE). Developed under the auspices of, and ratified by, IEEE, it defines an operating system interface upon which the client and service may (but need not) rely for their implementation.

- IEEE Std 1003.1-1988, IEEE Standard Portable Operating System Interface for Computer Environments, Institute of Electrical and Electronics Engineers, September 1988.

The following document supplements X.400 (1988). Developed under the auspices of, and ratified by, the CCITT, it provides postal delivery mode definitions for the MH packages.

- Recommendation F.170, Operational Provision for the International Public Facsimile Service Between Public Bureaux, CCITT Blue Book, Fascicle II.5, International Telecommunications Union, 1988.
- Recommendation T.411, see also ISO 8613-1.

Additional references:

- A/3311 EWOS/ETSI MHS Profile defined in CEN/CENELEC M-IT-02. (Also published as CEN/CENELEC ENV 41214.)
- NIST OSI Implementor's Workshop (OIW) Stable Implementors Agreement.

Chapter 1

Introduction

1.1 Purpose

This specification defines two application program interfaces (APIs) of architectural importance in the construction of message handling systems (MHSs). It defines an X.400 Application API that makes the functionality of a message transfer system (MTS) accessible to a message store (MS) or user agent (UA), or the functionality of a simple MS accessible to a UA. It also defines an X.400 Gateway API that divides a message transfer agent (MTA) into two software components, a mail system gateway and an X.400 gateway service.

The two interfaces support Message Access (MA), Message Transfer (MT), and Interpersonal Messaging (IM). MA is the conveyance of information objects between the MTS and one of its users, and thereby among its various users. MT is the conveyance of such objects between the MTAs that make up the MTS. IM is the MA and MT application in which the contents of messages are electronic memos. The interfaces address the operational requirements of these activities; they do not address their system management requirements, for example, those for security.

Note: Of the many applications of MA and MT, IM is singled out by this document only because it is singled out by X.400 itself. (X.400 highlights IM for purely historical reasons.) The use of the interfaces is not limited to IM. Since they permit the contents of messages to be arbitrary binary data, the interfaces are germane to all applications, for example, electronic data interchange (EDI).

Henceforth, the following terms are used consistently as indicated.

- The terms *MA interface* and *MT interface* denote the X.400 Application API and the X.400 Gateway API, respectively.
- The term *interface* denotes either interface without distinction, or one interface in particular. If the latter, which interface is meant will be clear from the context.
- The term *service* denotes software that implements an interface. A service that implements the MA interface realises the MTS. A service that implements the MT interface realises an X.400 gateway service.
- The term *client* denotes software that uses an interface. A client that uses the MA interface realises a UA or an MS. A client that uses the MT interface realises a mail system gateway.

  Note: The MA interface does not provide the functionality of X.400's MS Abstract Service (equivalently, its P7 protocol), although it can be viewed as providing a small portion of it. The functionality of P7 is outside the scope of this document. A mail system's message storage and retrieval capabilities are among its most important product-differentiating aspects. So that the interface does not effectively standardise those capabilities, their provision is made the responsibility of the client, not the service, and thus the design of those capabilities is left the responsibility of the client manufacturer.

1.2 Overview

The MA interface is high-level. It centres around three persistent databases that the service maintains even when it is out of contact with the client. The client places outbound messages and probes in a submission queue (the first database) for subsequent delivery or deliverability verification. The service places inbound messages and reports in a delivery queue (the second database) for subsequent delivery to the client. Alternatively, it places them in a retrieval queue (delivering them to the associated user in the process) for subsequent retrieval.

The MT interface also is high-level. It centres around two persistent databases which the service maintains even when it is out of contact with the client. The client places outbound messages, probes and reports in the output queue (the first database) for subsequent action by the service. The service places inbound messages, probes and reports in the input queue (the second database) for subsequent action by the client.

Both interfaces are explicit. They do not rely upon implicit communication channels between the client and the service. In particular, they neither presuppose nor exploit their use of a common file system or storage manager. They do assume, however, that the underlying operating system provides a means for signalling events (for efficient implementation of the *wait*( ) functions).

The interfaces are designed to be used and implemented in conjunction with the use and implementation of the general-purpose OM API defined in the referenced **XOM** Specification. Throughout this document, the term *OM interface* denotes the **OM API**.

1.3 Levels

This document defines the interfaces at two levels of abstraction. It defines *generic interfaces* independent of any particular programming language, and *C interfaces* for the variant of C standardised by the American National Standards Institute (ANSI). It does not define interfaces specific to other languages.

The C interface definitions provide language-specific declarations beyond the scope of the generic interface definitions. For readability alone, the specifications of the generic and C interfaces are physically combined, rather than physically separated.

1.4 Identifiers

How the identifier for an element of a C interface is derived from the name of the corresponding element of the generic interface depends upon the element's type as specified in Table 1-1. The generic name is prefixed with the character string in the second column of the table, alphabetic characters are converted to the case in the third column, and an underscore (_) is substituted for each hyphen (-) or space ( ).

Element Type	Prefix	Case
Data type	MH_	Lower
Data value	IM_, MH_, SM_	Upper
Data value (Class[1])	IM_C_, MH_C_, SM_C_	Upper
Data value (Value Length)	IM_VL_, MH_VL_, SM_VL_	Upper
Data value (Value Number)	IM_VN_, MH_VN_, SM_VN_	Upper
Data value component	none	Lower
Function	ma_, mt_	Lower
Function argument	none	Lower
Function result	none	Lower

[1] The Class Data Values with these prefixes denote the class variables of the OM String data type. The name of the C identifier denoting only the Elements component of the OM String data type of a Class Data Value is further prefixed by "OMP_O_.

Table 1-1 Derivation of C Identifiers

The prefix **mhP** is reserved for use by implementors of the service. The prefixes **mhX** and **MHX** are reserved for the proprietary extension of the interface. In all other respects, such extension is outside the scope of this document.

1.5 Object Management

The interface makes use of facilities provided by the OM API. These facilities are fully described in the referenced **XOM** Specification, and are introduced briefly below.

The subsections below introduce the various concepts which are used in Object Management, starting with the smallest.

1.5.1 Syntax

A *syntax* is the basis for the classification and representation of values in Object Management. Examples of syntaxes are Boolean, Integer, String(Octet) and Object.

Syntaxes are defined in the Object Management specification, and nowhere else, and are themselves represented by integers.

1.5.2 Value

A *value* is a single datum, or piece of information. Each value belongs to exactly one syntax by which its representation is defined. A value may be as simple as a Boolean value (for example, True), or as complicated as an entire OM object (for example, a Message).

1.5.3 OM Attribute

An *OM attribute type* is an arbitrary category into which a specification places some values.

OM attribute types are represented by integers, which are assigned in individual service specifications, and which are only meaningful within a particular package (see Section 1.5.6).

An *OM attribute* is an OM attribute type, together with an ordered sequence of one or more values. OM attributes can occur only as parts of an OM object and the OM attribute type and values are constrained by the OM class specification of that OM object (see Section 1.5.5).

The OM attribute type can be thought of as the name of the OM attribute.

There is no general representation for an OM attribute, but a descriptor represents an OM attribute type together with a single syntax and value (see Section 1.5.9).

1.5.4 OM Object

An *OM object* is a collection of OM attributes, the values of which can be accessed by means of functions. The particular OM attribute types which may occur in an OM object are determined by the OM class of the OM object (see Section 1.5.5), as are the constraints on those OM attributes. The OM class of an OM object is determined when the OM object is created, and cannot be changed.

OM objects are represented in the interface by a handle, or opaque pointer. The internal representation of an OM object is not specified though there is a defined data structure, called a *descriptor list*, which can also be used directly in a program (see Section 1.5.9).

1.5.5 OM Class

An *OM class* is a category of OM object, set out in a specification. It determines the OM attributes that may be present in the OM object, and details the constraints on those OM attributes.

Each OM object belongs directly to exactly one OM class, and is called an *instance* of that OM class.

The OM classes of OM objects form a tree. each OM class has exactly one immediate *superclass* (except for the OM class *Object*, which is the root of the tree), and each OM class may have an arbitrary number of *subclasses*. The tree structure is also known as the *OM class hierarchy*. The importance of the OM class hierarchy stems from the inheritance property which is discussed below.

Each OM class of OM object has a fixed list of OM attribute types and every OM object which is an instance of the OM class has only these OM attributes (actually some OM attributes may not be present in particular instances, as permitted by the constraints in the OM class specification). The list of OM attribute types which may appear in instances of an OM class has two parts. Each OM class *inherits* all the OM attribute types which are permitted in its immediate superclass as legal OM attribute types. There is also a list of additional OM attribute types that are permitted in the OM class. Any subclasses of this OM class will inherit all of these OM attribute types, from both lists.

Because of inheritance, an OM object is also said to be an instance of all its superclasses. It is required that the OM class constraints of each superclass are met, considering just those OM attribute types that are permitted in the superclass.

The OM class hierarchy and the list of OM attribute types for each OM class are determined solely by the interface specification and cannot be changed by a program.

The OM class specification may impose arbitrary constraints on the OM attributes. The most common of these are tabulated in the OM class specification and are marked with a * below. Frequently encountered cases include constraints to:

- restrict the syntaxes permitted for the values of an OM attribute (often to a single syntax) *
- restrict the particular values to a subset of those permitted by the syntax
- require exactly one value of the OM attribute (a *mandatory* OM attribute) *
- require either zero or one value of the OM attribute (an *optional* OM attribute) *
- permit multiple values, perhaps up to some limit known as the *value number constraint* *
- restrict the length of strings, up to a limit known as the *value length constraint*. *

Note: Any constraints on the length of a string are expressed in terms of bits in a bit string, octets in an octet string, or characters in a character string. These constraints are specified in the appropriate class definitions. However, the lengths of strings are everywhere else stated in terms of the number of *elements*, which are either bits or octets. The number of elements in a string with multibyte characters (for example, T.61 Teletext) may thus exceed the value length constraint. In C, an array with more bytes will be needed to store it.

The constraints may affect multiple OM attributes at once, for example a rule that only one of several OM attributes may be present in any OM object.

Every OM object includes the OM class to which it belongs as the single value of the mandatory OM attribute type **Class**, which cannot be modified. The value of this OM attribute is an OSI Object Identifier, which is assigned to the OM class by the specification.

An *abstract class* is an OM class of which instances are forbidden. It may be defined as a superclass in order to share OM attributes between OM classes, or simply to ensure that the OM class hierarchy is convenient for the interface definition.

1.5.6 Package

A *Package* is a set of OM classes which are grouped together by the specification, because they are functionally related.

A package is identified by an OSI Object Identifier, which is assigned to the package by the specification. Thus the identity of each package is completely unique.

1.5.7 Package Closure

An OM class may be defined to have an OM attribute whose value is an OM object of an OM class that is defined in some other package. This is done to share definitions and to avoid duplication. For example, the Message Handling package defined in Chapter 5 defines an OM class called **OR Name**. This OM class has an OM attribute whose value is an OM object of OM class **Name**, which is defined in the Directory Service package in the referenced **XDS** Specification. An OM class may also be a subclass of an OM class in another package. These relationships between packages lead to the concept of a Package-Closure.

A *Package-Closure* is the set of classes which need to be supported in order to be able to create all possible instances of all classes defined in the package. A formal definition is given in the referenced **XOM** Specification.

1.5.8 Workspace

Details of the representation of OM objects and of the implementation of the functions that are used to manipulate them are not specified, because they are not the concern of the application programmer. However, the programmer sometimes needs to be aware of which implementation is being used for a particular OM object.

The case of the OM class **Name** was mentioned above. This OM class is used in both the Message Transfer Service and in the Directory Service (defined in the referenced **XDS** Specification). If an application uses both services, and the two services use different internal representations of OM objects (perhaps because they are supplied by different vendors), then it is necessary for the application to specify which implementation should create a **Name** OM object. This is done by means of a workspace.

A *workspace* is one or more Package-Closures, together with an implementation of the Object Management functions which supports all the OM classes of OM objects in the Package-Closures.

The notion of a workspace also includes the storage used to represent OM objects and management of that storage. The interested reader should refer to the referenced **XOM** Specification for more details of how workspaces are implemented.

The application must obtain a workspace that supports an OM class before it is able to create any OM objects of that OM class. The workspaces are returned by functions in the appropriate service. For example, *mt_open*( ) returns a workspace that supports the Message Handling package, whilst another function in another OSI service returns a workspace that supports another package.

Some implementations may support additional packages in a workspace. For example, vendors may provide OM classes for additional security classes in the Secure Messaging package. (*mt_open*( ) may be used to negotiate these packages into the workspace.) Another important case is where two or more services are supported by the same implementation. In this case, the workspaces returned by the Directory Service function and the Message Transfer Service function *mt_open*( ) are likely to be the same one. The application need take no account of this, but may experience improved performance.

1.5.9 Descriptor

A *descriptor* is a defined data structure that is used to represent an OM attribute type and a single value. The structure has three components: a type, a syntax and a value.

A *descriptor list* is an ordered sequence of descriptors that is used to represent several OM attribute types and values.

Where the list contains several descriptors with the same OM attribute type (representing a multi-valued OM attribute), the order of the values in the OM attribute is the same as the order in the list. Such descriptors will always be adjacent.

Where the list contains a descriptor representing the OM class, this must occur before any others.

A *public object* is a descriptor list that contains all the OM attribute values of an OM object, including the OM class. Public objects are used to simplify application programs by enabling the use of static data structures instead of a sequence of OM function calls.

A *private object* is an OM object created in a workspace using the Object Management functions or the functions in an OSI service. The term is simply used for contrast with a public object.

1.5.10 Use of Objects

OM objects are used to represent the data collections used in the interface, such as parts of a message or messaging operation results. Refer to the chapters covering Interpersonal Messaging Packages (Chapter 4), Message Handling Packages (Chapter 5), and Secure Messaging Package (Chapter 6), for the definition of these OM classes.

An important feature of the interface is that an instance of a subclass can be used wherever a particular OM class is needed. This means both that the application can supply a subclass and that the service can return a subclass. For example, the application can submit messages in any format which is defined as a subclass of the class **Submitted Communique**, and the service returns all results of the *mh_submit*( ) function in any subclass of the **Submission Results** class.

Because the service may return a subclass of the specified OM class, applications should always use the *om_instance*( ) function when checking the OM class of an OM object, rather than testing the value of the **Class** OM attribute.

When the application supplies a subclass of the specified OM class as an argument, the service will either recognise them as vendor extensions or will ignore all OM attribute types which are not permitted in the specified OM class.

The application can generally supply either a public object or a private object as an argument of the interface functions. There are exceptions such as the **Session** argument, which must be a private object in the interests of efficiency. The interface will always return private objects. The application can convert these into public objects by a call to *0m_get*( ), if required.

Note that public objects returned by *om_get*( ) are read-only and must not be modified in any way.

1.6 Features

The interfaces enable the client and service to negotiate the use of the various defined features of the MA and MT interfaces. The *features* that the present edition of this document defines are the functional units (see below) and packages (see the referenced **XOM** Specification) that the document defines, and aspects of those functional units and packages identified as features.

A *functional unit* (FU) is a group of related functions. Other documents, or other editions of this document, may define (for example, proprietarily) additional features, which may (but need not) themselves be FUs or packages.

The features defined in the present edition of this document are identified in Table 1-2. The first column of the table identifies the feature. The second column gives the symbolic constant that names the object identifier that denotes the feature. The third column specifies the object identifier using ASN.1.

Feature	Symbol	Object Identifier (ASN.1)
Basic Access FU	basic-access	[iso(1) member-body(2) us(840) IEEE-P1224.1(10018) basic-access(1)]
Submission FU	submission	[iso(1) member-body(2) us(840) IEEE-P1224.1(10018) submission(2)]
Delivery FU	delivery	[iso(1) member-body(2) us(840) IEEE-P1224.1(10018) delivery(3)]
Retrieval FU	retrieval	[iso(1) member-body(2) us(840) IEEE-P1224.1(10018) retrieval(4)]
Basic Transfer FU	basic-transfer	[iso(1) member-body(2) us(840) IEEE-P1224.1(10018) basic-transfer(5)]
Transfer In FU	transfer-in	[iso(1) member-body(2) us(840) IEEE-P1224.1(10018) transfer-in(6)]
Transfer Out FU	transfer-out	[iso(1) member-body(2) us(840) IEEE-P1224.1(10018) transfer-out(7)]
IM 84 Package	im-84	[iso(1) member-body(2) us(840) IEEE-P1224.1(10018) im-84(8)]
IM 88 Package	im-88	[iso(1) member-body(2) us(840) IEEE-P1224.1(10018) im-88(9)]
MH 84 Package	mh-84	[iso(1) member-body(2) us(840) IEEE-P1224.1(10018) mh-84(10)]
MH 88 Package	mh-88	[iso(1) member-body(2) us(840) IEEE-P1224.1(10018) mh-88(11)]
SM 88 Package	sm-88	[iso(1) member-body(2) us(840) IEEE-P1224.1(10018) sm-88(12)]
Multiple Delivery[1]	multiple-delivery	[iso(1) member-body(2) us(840) IEEE-P1224.1(10018) multiple(13)]
General Content[2]	general-content	[iso(1) member-body(2) us(840) IEEE-P1224.1(10018) general-content(14)]

[1] The delivery of a message or report to several local recipients as the result of a single invocation of the *mh_start-delivery*( ) function. In the absence of this feature, the service confines itself to one recipient for each invocation.

[2] In certain circumstances a client may require access to the value of the message content octet string, for example, when engaged in secure messaging. If this feature is negotiated then the service will return a *General Content* attribute.

Table 1-2 Features and their Object Identifiers

The service makes a particular feature available to the client only if the client requests it. The client may request a set of features as a parameter to the *open*( ) function of the MA or MT interface. The service provides the Basic Access FU and Basic Transfer FU respectively as default for the two interfaces.

The following takes place if the client attempts to use a feature that the service has not made available.

- If the client invokes a function in the Basic Access or Basic Transfer FU, but that FU is unavailable, the service's behaviour is undefined.
- If the Basic Access FU is available and the client invokes a function in the Submission, Delivery or Retrieval FU, but that FU is unavailable, the feature-unavailable error arises.
- If the Basic Transfer FU is available and the client invokes a function in the Transfer In or Transfer Out FU, but that FU is unavailable, the feature-unavailable error arises.
- If the client supplies as a function argument an object, one of whose subobjects is an instance of a class that is not in an available package, the no-such-class error arises.

The following feature pairs are mutually exclusive for a particular session between the client and the service:

- the Basic Access FU and the Basic Transfer FU
- the Delivery FU and the Retrieval FU.

1.7 Options

A number of aspects of the service's behaviour are implementation-defined. These aspects are summarised as follows:

- the circumstances that define abnormal termination of a session
- the features of a session that the *open*( ) function supplies as defaults
- whether the service will accept any circumstances as causes of temporary delivery failure
- the maximum number of users that may be assigned to a single delivery queue
- the maximum number of sessions that may exist simultaneously
- which qualifying object in the delivery or input queue (if there are several such objects) the *start_delivery*( ) or *start_transfer_in*( ) function selects for delivery or transfer in
- which and how many sessions the *wait*( ) function notifies of an object's arrival when several sessions provide access to a delivery, retrieval or input queue.

1.8 Conformance

A manufacturer shall claim conformance to this specification only if it and its product collectively satisfy the following requirements:

- **Interfaces**
  The manufacturer and product shall satisfy the following requirements related to interfaces:
  - The manufacturer shall identify the interfaces (MA, MT, or both) the product implements, and state what roles (client, service, or both) it plays for each.
  - The product shall implement the OM interface as defined in the referenced **XOM** Specification, satisfying its conformance requirements, and play the same roles for that interface as it plays for the MA and MT interfaces.
- **Features**
  The manufacturer and product shall satisfy the following requirements related to features:
  - If the product plays the role of service for the MA or MT interface, the manufacturer shall state which features it implements.
  - If the product plays the role of service for the MA interface, it shall implement the Basic Access FU and the Submission FU, the Delivery FU, the Retrieval FU, or any combination of the three.
  - If the product plays the role of service for the MT interface, it shall implement the Basic Transfer FU and the Transfer In FU, the Transfer Out FU, or both.
  - If the product plays the role of service for either interface, the product shall implement the MH 88 Package, the MH 84 Package, or both.
- **Functions**
  The manufacturer and product shall satisfy the following requirement related to functions:
  - The product shall implement every aspect of every function in each FU for which it plays the role of service.
- **Classes**
  The manufacturer and product shall satisfy the following requirements related to classes:

— If the product implements the IM 88 or MH 88 Package, the manufacturer shall state which (if any) of its 1988 classes it implements.

— If the product implements the IM 88 or IM 84 Package, the manufacturer shall state which (if any) immediate subclasses of the Body Part class the product implements.

— If the product implements the IM 84 or SM 88 Package, it shall implement every class the package contains.

— If the product implements the MH 84 package, it shall implement all of those classes that are used by the functional units that it supports.

— The product shall implement the closures of all classes it implements.

— The product shall state what classes it provides the *encode*( ) function for.

- **Protocols**
  The manufacturer and product shall satisfy the following requirements related to protocols:

  — If the product plays the role of service for the MA interface, the manufacturer shall state whether or not it realises the interface by means of X.400's Message Submission and Delivery Protocol (P3) or Message Retrieval Protocol (P7).

  — If the product plays the role of service for the MT interface, the manufacturer shall state whether or not it realises the interface by means of X.400's Message Transfer Protocol (P1).

  — If the product plays the role of service for either interface, and implements the IM 88 or IM 84 Package, the manufacturer shall state whether or not the product realises the interface by means of X.400's Interpersonal Messaging Protocol (P2).

  — If the product implements P1 (1988), P3 (1988) or P7 (1988), the manufacturer and product shall satisfy the conformance requirements of X.400 (1988) with respect to those protocols.

  Note: A 1988 package can be realised by means of the corresponding 1988 protocol alone, but a 1984 package can be realised using either the 1984 or the 1988 protocol.

- **Options**
  The manufacturer and product shall satisfy the following requirement related to implementation:

  — If the product plays the role of service for the MA or MT interface, the manufacturer shall state the behaviour of implementation-defined options.

- **Interpretation of "any" syntax**

  Wherever an *any* appears in the syntax column of an attribute definition, this shall be treated as the corresponding OM syntax wherever the underlying ASN.1 encoding is a `Universal` simple type as listed in the referenced **XOM** Specification, and as `String(Encoding)` otherwise.

1.9 Abbreviations

The following abbreviations are used throughout this document.

ADMD	administration management domain
ANSI	American National Standards Institute
API	application program interface
ASN.1	Abstract Syntax Notation One
AU	access unit
BER	Basic Encoding Rules
C	conditional
CA	certification authority
CCITT	International Telegraph and Telephone Consultative Committee
CDS	Command Document Start
DCS	Digital Command Signal
DDA	domain-defined attribute
DL	distribution list
DR	delivery report
DS	directory system
EDI	electronic data interchange
EIT	encoded information type
FIF	Facsimile Information Field
FU	functional unit
G3	Group 3 (facsimile)
G4	Group 4 (facsimile)
IA5	International Alphabet No. 5
IEEE	Institute of Electrical and Electronics Engineers
IM	Interpersonal Messaging
IPM	interpersonal message
IPN	interpersonal notification
ISDN	Integrated Services Digital Network
ISO	International Organisation for Standardisation
ITA2	International Alphabet No. 2
M	mandatory
MA	Message Access
MD	management domain

MH	Message Handling
MHS	message handling system
MOTIS	Message Oriented Text Interchange System
MS	message store
MT	Message Transfer
MTA	message transfer agent
MTS	message transfer system
NBP	non-basic parameter
NDR	non-delivery report
NRN	non-receipt notification
O/R	originator/recipient
OM	Object Management
OSI	Open Systems Interconnection
P1	Message Transfer Protocol
P2	Interpersonal Messaging Protocol
P3	Message Submission and Delivery Protocol
P7	MS Access Protocol
PAEK	public asymmetric encryption key
PDAU	physical delivery access unit
PDS	physical delivery system
POSIX	Portable Operating System Interface for Computer Environments
PRMD	private management domain
RD	recipient descriptor
RN	receipt notification
SAEK	secret asymmetric encryption key
SFD	simple formattable document
SM	Secure Messaging
UA	user agent

Chapter 2

Functional Architecture

2.1 Introduction

This chapter describes the overall functional architecture of the systems whose construction the MA and MT interfaces are intended to facilitate. As previously indicated, the MA interface either makes the functionality of the MTS accessible to an MS or a UA, or makes the functionality of a simple MS accessible to a UA. The MT interface, on the other hand, divides an MTA into two software components.

2.2 Message Access Interface

This section describes the MA interface. It gives a conceptual model, a functional overview, and the division of responsibility between the client and the service.

2.2.1 Conceptual Model

As depicted in Figure 2-1, the functions of the MA interface manipulate queues of three kinds (submission, delivery and retrieval) which the service maintains for the client in non-volatile storage. The service maintains these queues even when it is out of contact with the client. Thus the service is active, not passive.

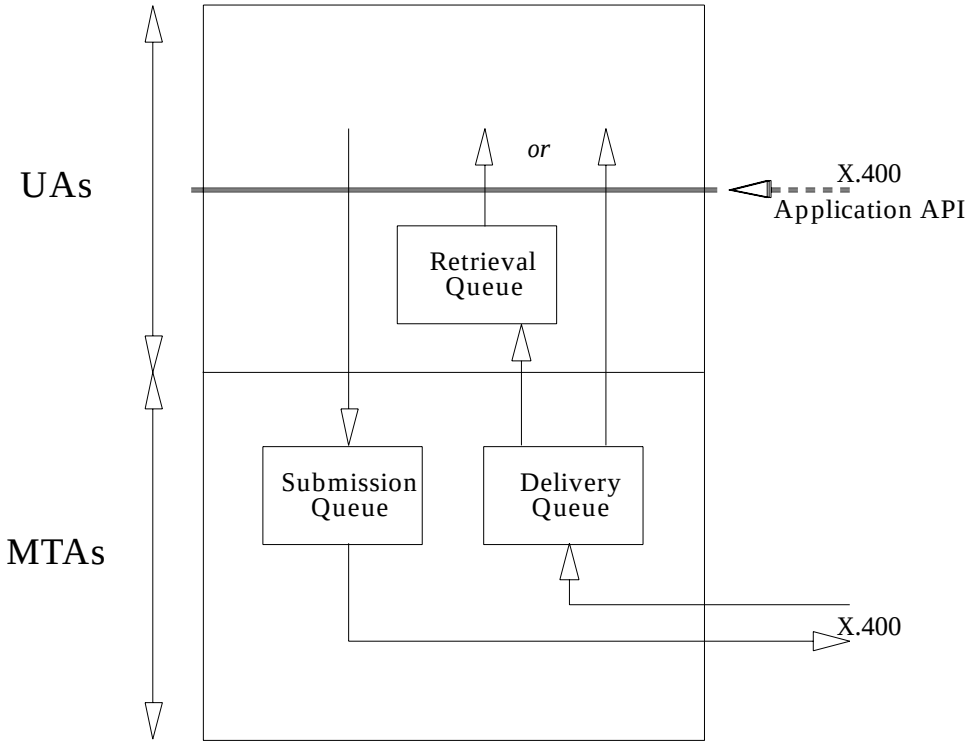

Figure 2-1 Conceptual Model of the X.400 Application API

The service maintains a submission queue and either a delivery queue or a retrieval queue for each local user, except that a single delivery queue may be shared by a group of local users (see the definition of the *open* function in Chapter 3). The choice, for each user, between a delivery or retrieval queue is communicated to the service by means outside the scope of this document, as is the assignment of users to delivery queues.

Note: One realisation of the service involves the use of the (low-level) delivery queue as a staging ground for messages and reports bound for a (higher-level) retrieval queue. In such a realisation, a single user can be said to have both a delivery queue and a retrieval queue. But this state of affairs, even if it exists, is not visible to the client through the interface.

A user's *submission queue* is filled by the client and emptied by the service. At any point in time, it comprises zero or more objects to be conveyed by the service in the direction of their intended recipients. Each such object is a message or a probe. The service processes objects in the queue, in part, on the basis of their priority. The addition of an object to the queue represents a transfer of responsibility for the object from the client to the service (submission). The addition does not significantly delay the client, being independent of any actions the service may be taking concurrently with respect to objects previously added to the queue.

Note: At any moment, either the client or the service (but not both) has responsibility for a message, probe or report, that is, it has decided whether it must be placed in non-volatile storage and, if it must, has placed it there.

A user's or user group's *delivery queue* is filled by the service and emptied by the client. At any point in time, it comprises zero or more objects to be conveyed by the client to one or more of the users associated with the queue. Each such object is a message or a report. The service positions objects in the queue, in part, on the basis of their priority. The removal of an object from the queue represents a transfer of responsibility for the object from the service to the client (delivery).

A user's *retrieval queue* is filled by the service and emptied by the client. At any point in time, it comprises zero or more objects to be conveyed by the client to the user. Each such object is a message or a report. The service positions objects in the queue, in part, on the basis of their priority. The addition of an object to the queue represents a transfer of responsibility for the object (delivery). The removal of the object from the queue represents another transfer of responsibility for the object, from the service to the client (retrieval).

The client can access an object in a delivery or retrieval queue without first removing it from that queue, yet with confidence that no other client or client instance can access the object simultaneously. The object is said to be *reserved* by the client. Whenever the client requests such access to an object, the service considers only those objects not already reserved. If all the objects are reserved, the request is denied. Reservations do not survive the termination, normal or abnormal, of the session (see Section 2.2.2) in which they are made, but rather are terminated by the service. The circumstances that define abnormal termination of a session are system-defined.

2.2.2 Functional Overview

The MA interface comprises functions that enable the client to manipulate a user's submission queue, thereby conveying messages and probes to the service, and its delivery or retrieval queue, thereby receiving messages and reports from the service. The client uses the OM interface to create messages and probes for submission and to examine and perhaps modify delivered or retrieved messages and reports.

One of two principal tasks of the typical client is to submit messages and (perhaps) probes that functionally approximate objects in its local environment. To do this in a particular instance, the client establishes a session with the service using the *open*( ) function, creates a new object using the *om_create*( ) function, places the local information in it using the *om_put*( ) and *om_write*( ) functions, submits the object using the *submit*( ) function, and terminates the session using the *close*( ) function.

The second principal task of the typical client is to inject into the local environment information drawn from delivered messages and reports. To do this in a particular instance, the client establishes a session with the service using the *open*( ) function, awaits the next inbound object using the *wait*( ) function, obtains access to it using the *start_delivery*( ) or *start_retrieval*( ) function, draws information from it using the i.Fn om_get and *om_read*( ) functions and injects that information into the local environment, accepts responsibility for the object using the Fn

finish_delivery or *finish_retrieval*( ) function, and terminates the session using the *close*( ) function.

The above scenarios are exemplary, not exhaustive. The MA and OM interfaces provide other functions that enable more complex client-service interactions. The client, for example, can determine how many objects await delivery or retrieval using the *size*( ) function, cancel a delivery or retrieval using the *finish_delivery*( ) or *finish_retrieval*( ) function, copy an object using the *om_copy*( ) function, and modify an object using, for example, the *om_remove*( ) function.

The client and service interact only in the context of a session (see Section 5.3.42 on page 133).

2.2.3 Division of Responsibility

The MA interface, to a great extent, represents the boundary between a UA (the client), and its MTA (the service). The submission and delivery queues and the functions surrounding them embody the operational (but not the administrative) functionality of the MTS Abstract Service of X.411 (1988). The retrieval queues and the functions surrounding them provide additional functionality (a simple *hold for retrieval* capability), having no counterpart in X.400.

Notes:

1. Alternatively, a retrieval queue may be viewed as providing a small portion of the operational functionality of the MS Abstract Service of X.413 (1988).
2. The service may also provide some or all of the administrative or management functionality of an MTA, but this is outside the scope of this document.

Besides maintaining the client's submission, delivery and retrieval queues, the service shall:

- formally submit messages and probes to the MTA at the client's request
- formally take delivery of messages and reports from the MTA at the client's request, when a delivery queue is used
- formally take delivery of messages and reports from the MTA asynchronously, when a retrieval queue is used.

Optionally, the service may also:

- implement P2
- implement P3
- implement P7
- implement the lower-layer protocols for OSI dictated by X.400
- actively establish and terminate OSI connections with, and passively accept connections from, a designated MS or MTA
- send and receive requests and responses via OSI connections
- encode and decode requests and responses in accordance with the Basic Encoding Rules (BER) of ASN.1.

Note: The interface is designed so that the service can implement it by means of the P3 or P7 protocol, although this is not the only, or even the most likely implementation approach. It does not follow from this, however, that every aspect of the interface has a counterpart in P3 or P7. In particular, the interface's ability to sustain a single session on behalf of two or more users has no analogue in P3.

Several tasks are left to the client. The client must:

- place objects in the submission queue as it desires
- remove objects from the delivery queue in a timely fashion, if a delivery queue is used
- remove objects from the retrieval queue at its leisure, if a retrieval queue is used
- convert outbound objects from, and inbound objects to, the client's native format.

Notes:

1. The service's fulfillment of some of the above responsibilities may be subject to local management policies which are outside the scope of this document.
2. Nothing above is intended to prevent the service from communicating with MTAs and MSs by non-OSI means, either instead of or in addition to communicating with them using OSI protocols. If the former is done, the service need not implement the OSI protocols mentioned above.

2.3 Message Transfer Interface

This section describes the MT interface. It gives a conceptual model, a functional overview, and the division of responsibility between the client and the service.

2.3.1 Conceptual Model

As depicted in Figure 2-2, the functions of the MT interface manipulate queues of two kinds (output and input) which the service maintains for the client in non-volatile storage. The service maintains these queues even when it is out of contact with the client. Thus the service is active, not passive.

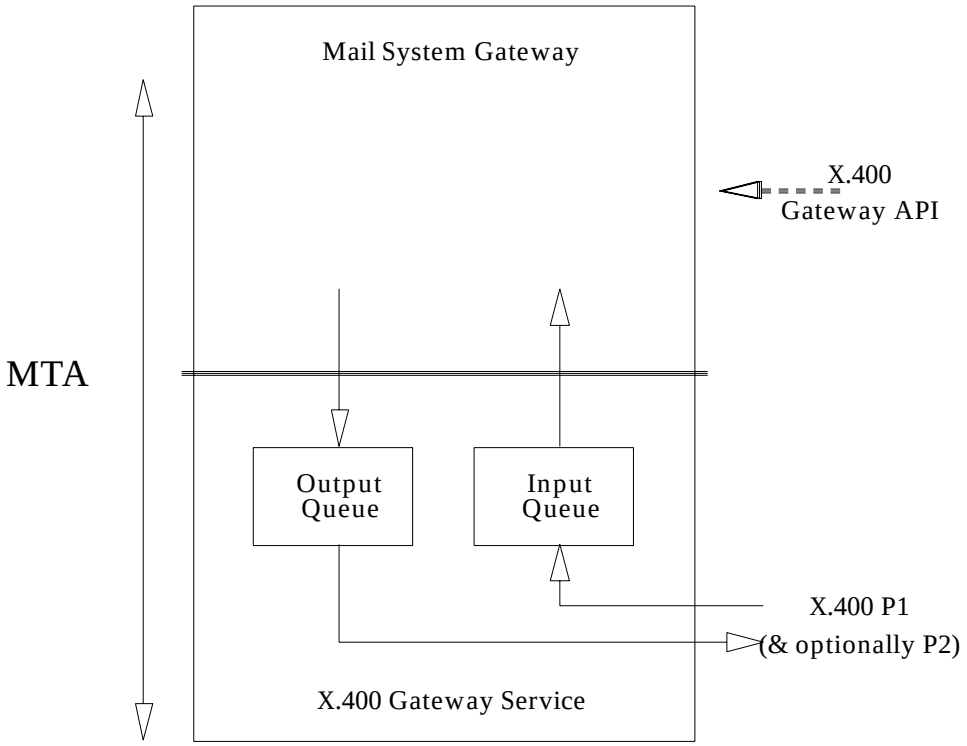

Figure 2-2 Conceptual Model of the X.400 Gateway API

The service maintains an output queue and an input queue for each client.

A client's *output queue* is filled by the client and emptied by the service. At any point in time, it comprises zero or more objects to be conveyed by the service in the direction of their intended recipients. Each such object is a message, a probe, or a report. The service processes objects in the queue, in part, on the basis of their priority. The addition of an object to the queue represents a transfer of responsibility for the object from the client to the service. The addition does not significantly delay the client, being independent of any actions the service may be taking concurrently with respect to objects previously added to the queue.

A client's *input queue* is filled by the service and emptied by the client. At any point in time, it comprises zero or more objects to be conveyed by the client in the direction of users of the mail system to which it is a gateway. Each such object is a message, a probe or a report. The service positions objects in the queue, in part, on the basis of their priority. The removal of an object

from the queue represents a transfer of responsibility for the object from the service to the client.

The client can access an object in its input queue without first removing it from that queue, yet with confidence that no other client or client instance can access the object simultaneously. The object is said to be *reserved* by the client. Whenever the client requests such access to an object, the service considers only those objects not already reserved. If all the objects are reserved, the request is denied. Reservations do not survive the termination (normal or abnormal) of the session (see Section 2.3.2) in which they are made, but rather are terminated by the service. The circumstances that define abnormal termination of a session are system-defined.

2.3.2 Functional Overview

The MT interface comprises functions that enable the client to manipulate its output queue, thereby conveying messages, probes and reports to the service, and its input queue, thereby receiving messages, probes and reports from the service. The client uses the OM interface to create objects for transfer out and to examine and perhaps modify objects that it is transferring in.

One of two principal tasks of the typical client is to transfer out messages, reports and (perhaps) probes that functionally approximate objects in its local environment. To do this in a particular instance, the client establishes a session with the service using the *open*( ) function, creates a new object using the *0m_create*( ) function, places the local information in it using the *om_put*( ) and *om_write*( ) functions, transfers out the object using the *transfer_out*( ) function, and terminates the session using the *close*( ) function.

The second principal task of the typical client is to inject into the local environment messages, reports and (perhaps) probes that functionally approximate objects transferred in. To do this in a particular instance, the client establishes a session with the service using the *open*( ) function, awaits the next inbound object using the *wait*( ) function, obtains access to it using the *start_transfer_in*( ) function, draws information from it using the *om_get*( ) and *om_read*( ) functions and injects that information into the local environment, accepts responsibility for the object using the *finish_transfer_in*( ) function, and terminates the session using the *close*( ) function.

The above scenarios are exemplary, not exhaustive. The MT and OM interfaces define other functions that enable more complex client-service interactions. The client, for example, can determine how many objects await transfer in using the *size*( ) function, cancel a transfer in using the *finish_transfer_in*( ) function, copy an object using the *om_copy*( ) function, and modify an object using, for example, the *om_remove*( ) function.

The client and service interact only in the context of a session (see Section 5.3.42 on page 133).

2.3.3 Division of Responsibility

The MT interface represents a boundary within an MTA, that is, the client and service together provide the functionality of an MTA. The interface divides that functionality between them in order to maximise the tasks the service performs, and thus minimise the tasks the client must perform. The service fulfills many of its responsibilities asynchronously with respect to its interactions with the client.

Nothing in this section is intended to prevent the service from communicating with MTAs by non-OSI means, either instead of or in addition to communicating with MTAs using OSI protocols. If the former, the service need not implement any OSI protocols. If the latter, besides maintaining the client's input and output queues, the service shall:

- optionally, implement P2
- implement P1
- implement the lower-layer protocols for Open Systems Interconnection (OSI) dictated by X.400
- maintain routing tables that identify zero or more *neighbouring MTAs*, that is, MTAs with which the service is prepared to establish OSI connections
- actively establish and terminate OSI connections with, and passively accept connections from, neighbouring MTAs
- send and receive objects (messages, probes and reports) via OSI connections
- encode and decode objects in accord with the BER
- route objects that come into its possession, replicating them, adding trace information, detecting routing loops, and generating non-delivery reports (NDRs), as required
- assign MTS identifiers to, and route to their destinations, any objects the client places in its output queue
- place in the client's input queue any objects that come into the service's possession, among whose destinations are those associated with the client by local management policy
- transfer objects among adjacent MTAs and other clients in a manner transparent to the client.

Several tasks are left to the client. The client must:

- place in the output queue any objects emitted by the mail system to which it is a gateway
- inject into that mail system any objects the service places in the input queue
- generate and place in the output queue any reports that may be provoked by the messages and probes it finds in the input queue
- convert outbound objects from, and inbound objects to, the mail system's native format.

Notes:

1. The service's fulfillment of some of the above responsibilities may be subject to local management policies which are outside the scope of this document.
2. Because the client's responsibilities are minimised and the service's responsibilities maximised, the MT interface can be characterised as high-level, rather than low-level. While the service is generic, the client is mail system-specific. A high-level interface is chosen to simplify the software that must be written once per mail system (that is, the client implementation) by complicating, of necessity, the software that must be written only once (that is, the service implementation). An interface that directly embodies the functionality of the MTA Abstract Service of X.411 (1988) would not provide the above commercial advantage and, in any case, is not dictated by OSI. The purpose of the abstract service definition is to prescribe the externally visible behaviour of an MTA, not dictate the modularity of an MTA implementation.

Chapter 3

Message Handling Interfaces

3.1 Summary

This chapter defines the MA and MT interfaces. It specifies the functions that the service makes available to the client, the data types of which the arguments and results of the functions are data values, and the return codes that denote the outcomes (in particular, the exceptions) that the functions may report. The chapter also summarises the declarations that define the C interfaces.

3.2 Data Types

This section defines, and the following table lists, the data types of the MA and MT interfaces that are specific to MH. The data types of both the generic and C interfaces are specified. Those of the C interface are repeated in Section 3.6 on page 56, which serves as a summary and a reference. The interfaces also include the Boolean, Object, Object Identifier, Private Object, Return Code, String and intermediate data types of the OM interface.

Data Type	Description
Feature	The features to be negotiated for a session.
Interval	An interval of time measured in milliseconds.
Object Count	A number of objects.
Sequence Number	The sequence number of an object in a retrieval queue.

Table 3-1 Interface Data Types Specific to MH

3.2.1 Feature

NAME

Feature - type definition for requesting features

C DECLARATION

```
typedef struct {
    OM_object_identifier    feature;
    OM_boolean              activated;
} MH_feature;
```

DESCRIPTION

A data value of this type is used for negotiating the features of a session.

3.2.2 Interval

NAME

Interval - the integer that denotes an interval of time measured in milliseconds

C DECLARATION

```
typedef OM_uint32 MH_interval;
```

DESCRIPTION

A data value of this data type is the integer in the interval $[0, 2^{32})$ that denotes an interval of time measured in milliseconds.

3.2.3 Object Count

NAME

Object Count - the integer that denotes a number of objects

C DECLARATION

```
typedef OM_uint32 MH_object_count;
```

DESCRIPTION

A data value of this data type is the integer in the interval $[0, 2^{32})$ that denotes a number of objects.

3.2.4 Sequence Number

NAME

Sequence Number - the sequence number of an object in a retrieval queue

C DECLARATION

```
typedef OM_uint32 MH_sequence_number;
```

DESCRIPTION

A data value of this data type is the integer in the interval [0, 2^{31}) (*sic*) that denotes a message or report in a retrieval queue.

Sequence numbers are assigned in ascending order, but not necessarily consecutively. An object's sequence number never changes, and no sequence number denotes two different objects, even at different times.

3.3 Access Functions

This section defines, and the following table lists, the functions of the MA interface. The functions of both the generic and C interfaces are specified. Those of the C interface are repeated in Section 3.6 on page 56, which serves as a summary and a reference.

Function	Description
Cancel Submission	Cancel a submitted message, with the deferred delivery option
Close	Terminate an MA session.
Finish Delivery	Conclude the delivery in progress in a session.
Finish Retrieval	Conclude the retrieval in progress in a session.
Open	Establish an MA session.
Size	Determine the size of the delivery or retrieval queue.
Start Delivery	Begin the delivery of a message or a report.
Start Retrieval	Begin the retrieval of a message or a report.
Submit	Submit a communique.
Wait	Return when an object is available for delivery or retrieval.

Table 3-2 MA Interface Functions

As indicated in Table 3-2, the MA interface comprises a number of functions whose purpose and range of capabilities are summarised as follows:

Cancel Submission
This function cancels a submitted message, with the deferred delivery option.

Close
This function terminates an MA session between the client and the service. If the delivery or retrieval of a message or a report is in progress, the service first unsuccessfully finishes that delivery or retrieval.

Finish Delivery
This function concludes the delivery in progress in a session. The client supplies delivery confirmations, as required, for users to which the object, a message, was delivered. It also indicates to which users the object, either a message or a report, is undeliverable.

Finish Retrieval
This function concludes the retrieval in progress in a session. The client indicates whether the service is to remove the retrieved message or report from the retrieval queue, or leave it there.

Open
This function establishes an MA session between the client and the service, and makes the Basic Access FU and the OM Package initially available in that session. The client may also specify the other features required for the session. The client specifies either its own name or the O/R address of a local user. The session provides MTS access to the local user at the specified address, if the latter, or to a group of local users, statically associated with the client name, if the former.

Size
This function determines the number of unreserved objects in the input queue to which a session provides access. Each object is a message or a report.

Start Delivery
This function begins the delivery of a message or a report to one or more of the users associated with a session by reserving an unreserved object in the delivery queue. If no messages or reports await delivery, the function reports an exception.

Start Retrieval

This function begins the retrieval of a message or a report by reserving an unreserved object in the retrieval queue to which a session provides access. If no messages or reports await retrieval, the function reports an exception.

Submit

This function submits a communique (see Section 5.3.5 on page 91) by adding it to the submission queue to which a session provides access. The function first verifies the communique's integrity.

Wait

This function returns when a message or a report is available for delivery or retrieval in the delivery or retrieval queue to which a session provides access, or when a period of time elapses, whichever occurs first.

The functions are grouped into four FUs (one basic and one each for submission, delivery and retrieval) as indicated in the following table. As stated previously, the Delivery and Retrieval FUs are mutually exclusive.

Basic Access	Submission	Delivery	Retrieval
Open Close OM API	Submit Cancel Submission	Size Start Delivery Finish Delivery Wait	Size Start Retrieval Finish Retrieval Wait

Table 3-3 MA Interface Functional Units

Note: The OM API is defined in the referenced **XOM** Specification.

The intent of the interface definition is that each function is atomic, that is, it either carries out its assigned task in full and reports success, or fails to carry out even a portion of the task and reports an exception. However, the service does not guarantee that a task will not occasionally be carried out in part but not in full.

Note: Making such a guarantee might be prohibitively expensive.

Whether a function detects and reports each of the exceptions listed in the Errors clause of its specification is unspecified. If a function detects two or more exceptions, which it reports is unspecified. If a function reports an exception for which a return code is defined, however, it uses that (rather than another) return code to do so.

NAME

cancel submission - cancel a submitted message, with the deferred delivery option

SYNOPSIS

```
#include <xmh.h>

OM_return_code
ma_cancel_submission (
    OM_private_object    session,
    OM_object            mts_identifier
);
```

DESCRIPTION

This function cancels the deferred delivery of a message, without regard to the session in which it was submitted.

ARGUMENTS

Session (Private Object)
: An established MA session between the client and the service; an instance of the Session class.

MTS Identifier (Object)
: The MTS identifier assigned to the message whose delivery is to be cancelled; an instance of the MTS Identifier class.

RESULTS

Return Code (Return Code)
: Whether the function succeeded and, if not, why. It may be **success** or one of the values listed under Errors below.

ERRORS

feature-unavailable, function-interrupted, memory-insufficient, network-error, no-such-message, no-such-session, no-such-syntax, no-such-type, not-private, permanent-error, pointer-invalid, system-error, temporary-error, too-late, too-many-values, wrong-class, wrong-value-length, wrong-value-makeup, wrong-value-number, wrong-value-syntax, wrong-value-type.

NAME

close - terminate an MA session between the client and the service

SYNOPSIS

```
#include <xmh.h>

OM_return_code
ma_close (
    OM_private_object    session
);
```

DESCRIPTION

This function terminates an MA session, as well as destroying the workspace, between the client and the service. If the delivery or retrieval of a message or a report is in progress in the session, the service first unsuccessfully finishes that delivery or retrieval.

ARGUMENTS

Session (Private Object)

An established MA session between the client and the service; an instance of the Session class.

RESULTS

Return Code (Return Code)

Whether the function succeeded and, if not, why. It may be **success** or one of the values listed under Errors below.

ERRORS

function-interrupted, memory-insufficient, network-error, no-such-session, not-private, permanent-error, pointer-invalid, system-error, temporary-error, wrong-class.

NAME

finish-delivery - conclude the delivery in progress in a session

SYNOPSIS

```
#include <xmh.h>

OM_return_code
ma_finish_delivery (
    OM_private_object    session,
    OM_object            delivery_confirmations,
    OM_object            non_delivery_reports
);
```

DESCRIPTION

This function concludes the delivery in progress in a session. The client should indicate to which users the object, if a message, or user, if a report, is undeliverable, by supplying Non-delivery reports. The client may optionally supply delivery confirmations, for users to which the object, a message was delivered. If the client does not explicitly confirm delivery or non-delivery, for all of the recipients on a message then the service assumes that the message has been delivered to the recipients not covered by a confirmation or non-delivery.

The client indicates to which users the message, or user, the report is temporarily, rather than permanently, undeliverable. The circumstances that cause temporary undeliverability are client-defined. However, whether the service will accept any circumstances as causes of temporary failure is service implementation-defined. If the service does not support temporary failures, it treats them as permanent.

If the object is a message, the service issues delivery reports (DRs), as required, for the users to which it has been delivered; issues NDRs, as required, for the users to which it is permanently undeliverable; and returns the message to the delivery queue for the users to which it is temporarily undeliverable. In the first case, the service considers that it has transferred responsibility for the message to the client. In the last case, the service marks the object unreserved.

If the object is a report and the user to which it was to be delivered is temporarily undeliverable, the service returns it to the delivery queue. If the user takes delivery of the report then the service considers that it has transferred responsibility for the report to the client.

In all cases, the object handle remains valid on return from the function and must be deleted by the client when no longer required.

The client should not invoke the OM Delete function on the object prior to calling this function; otherwise the behavior of this function is undefined.

ARGUMENTS

Session (Private Object)

An established MA session between the client and the service; an instance of the Session class. A delivery shall be in progress.

Delivery Confirmations (Object)

One or more delivery confirmations, as required, for users to which the object, a message, was delivered; an instance of the Local Delivery Confirmations class.

This argument may be omitted.

In the C interface, the argument's absence is signalled by the null pointer.

Non-delivery Reports (Object)
Indicates the one or more users to which the object, a message, cannot be delivered, or the single user to which the object, a report, cannot be delivered; an instance of the Local NDR class.

This argument is omitted if there are no such users.

In the C interface, the argument's absence is signalled by the null pointer.

RESULTS

Return Code (Return Code)
Whether the function succeeded and, if not, why. It may be **success** or one of the values listed under Errors below.

ERRORS

feature-unavailable, function-interrupted, memory-insufficient, network-error, no-such-object, no-such-session, no-such-syntax, no-such-type, not-private, permanent-error, pointer-invalid, session-not-busy, system-error, temporary-error, too-many-values, wrong-class, wrong-value-length, wrong-value-makeup, wrong-value-number, wrong-value-syntax, wrong-value-type.

NAME

finish-retrieval - conclude the retrieval in progress in a session

SYNOPSIS

```
#include <xmh.h>

OM_return_code
ma_finish_retrieval (
  OM_private_object   session,
  OM_boolean          remove
);
```

DESCRIPTION

This function concludes the retrieval in progress in a session. The client indicates whether the service is to remove the message or report from the retrieval queue or leave it there.

In the former case, the service considers that is has transferred responsibility for the objects from the service to the client; the object handles remain accessible on return from this function and may be deleted (using the *om_delete*( ) function) when no longer required.

In the latter case, the objects are returned to the input queue and marked unreserved. The associated object handles are made inaccessible on return from this function and should not be deleted by the client; however, the associated communique or report can be obtained again by a subsequent *start_transfer_in*( ) function invocation.

The client should not invoke the OM Delete function on the object prior to calling this function; otherwise the behavior of this function is undefined.

ARGUMENTS

Session (Private Object)
: An established MA session between the client and the service; an instance of the Session class. A retrieval shall be in progress.

Remove (Boolean)
: Whether the service is to remove the retrieved message or report from the retrieval queue, rather than leave it there.

RESULTS

Return Code (Return Code)
: Whether the function succeeded and, if not, why. It may be **success** or one of the values listed under Errors below.

ERRORS

feature-unavailable, function-interrupted, memory-insufficient, network-error, no-such-object, no-such-session, not-private, permanent-error, pointer-invalid, session-not-busy, system-error, temporary-error, wrong-class.

NAME

open - establish an MA session between the client and the service

SYNOPSIS

```
#include <xmh.h>

OM_return_code
ma_open (
  OM_public_object   user_address,
  OM_string          client_name,
  MH_feature         feature_list[],
  OM_private_object  *session,
  OM_workspace       *workspace
);
```

DESCRIPTION

This function establishes an MA session between the client and the service, and makes the Basic Access FU and the OM Package initially available in that session. The client may also specify the other features required for the session.

The client specifies either its own name or the O/R address of a local user. The session provides MTS access to the local user at the specified address, if the latter, or to a group of local users, statically associated with the client name, if the former. The Retrieval FU can be requested only if a single user is designated, but that user may be designated in either of these two ways.

The client always designates a particular user in the same way. The choice between O/R address and client name is made by means outside the scope of this document. How users are associated with a client name also is outside the document's scope. The maximum number of users in a group is implementation-defined (and may be one).

Opening an MA session also creates a workspace. A workspace contains objects returned as a result of functions invoked within that session. The workspace is used as an argument in the *om_create*( ) and *copy*( ) functions.

The maximum number of sessions that may exist simultaneously is implementation-defined and may vary with time.

ARGUMENTS

User Address (Object)

Explicitly identifies the local user to which the session is to provide MTS access; an instance of the OR Address class.

If this argument is absent, the Client Name argument shall be present.

In the C interface, the argument's absence is signalled by the null pointer.

Client Name (String)

The name by which the service knows the client, interpreted as a value whose syntax is String (IA5). It implicitly identifies one or more local users to which the session is to provide MTS access.

If this argument is absent, the User Address argument shall be present.

In the C interface, the argument's absence is signalled by a zero length string.

Feature-List (Feature-List)
An ordered sequence of features, each represented by an object identifier. The sequence is terminated by an object identifier having no components (a length of zero and any value of the data pointer in the C representation).

RESULTS

Return Code (Return Code)
Whether the function succeeded and, if not, why. It may be **success** or one of the values listed under Errors below.

Activated (Boolean-List)
If the function completed successfully, this result contains an ordered sequence of Boolean values, with the same number of elements as the Feature-List. If true, each value indicates that the corresponding feature is now part of the interface. If false, each value indicates that the corresponding feature is not available.

In the C binding, this result is combined with the Feature-List argument as a single array of structures of type **MH_feature**.

Session (Private Object)
The established MA session between the client and the service; an instance of the Session class. The service prevents the client from modifying this object subsequently. This result is present if, and only if, the Return Code result is success.

Workspace (OM_workspace)
The workspace that will contain all objects returned as a result of the functions invoked in the session.

ERRORS

feature-conflicts, feature-unavailable, function-interrupted, memory-insufficient, network-error, no-such-client, no-such-syntax, no-such-type, no-such-user, permanent-error, pointer-invalid, system-error, temporary-error, too-many-sessions, too-many-values, wrong-class, wrong-value-length, wrong-value-makeup, wrong-value-number, wrong-value-syntax, wrong-value-type.

NAME

size - determine the number of messages and reports in the delivery or retrieval queue

SYNOPSIS

```
#include <xmh.h>

OM_return_code
ma_size (
    OM_private_object    session,
    MH_object_count     *number
);
```

DESCRIPTION

This function determines the number of messages and reports in the delivery or retrieval queue to which a session provides access.

ARGUMENTS

Session (Private Object)

An established MA session between the client and the service; an instance of the Session class.

RESULTS

Return Code (Return Code)

Whether the function succeeded and, if not, why. It may be **success** or one of the values listed under Errors below.

Number (Object Count)

The number of unreserved objects in the delivery or retrieval queue. However, if that number exceeds 2^{16}-1, 2^{16}-1 is returned. This result is present if, and only if, the Return Code result is success.

ERRORS

feature-unavailable, function-interrupted, memory-insufficient, network-error, no-such-session, not-private, permanent-error, pointer-invalid, system-error, temporary-error, wrong-class.

NAME

start-delivery - begin the delivery of a message or a report to one or more of the users associated with a session

SYNOPSIS

```
#include <xmh.h>

OM_return_code
ma_start_delivery (
    OM_private_object    session,
    OM_private_object   *object
);
```

DESCRIPTION

This function begins the delivery of a message or a report to one or more of the users associated with a session (see the definition of the *open*( ) function) by reserving an unreserved object in the delivery queue. If no objects await delivery, the function reports an exception. The client shall finish the delivery of one object before it starts the delivery of another in the same session. The delivery of a particular object cannot simultaneously be in progress in two sessions.

Whether the service begins the delivery of an object addressed to several users in one function invocation, or in one invocation per user, is dependent on the Multiple-delivery feature being available for the session.

Which qualifying object in the delivery queue (if there are several such objects) the service selects for delivery is implementation-defined.

Notes:

1. The invocation of this function initiates, but does not consummate, delivery; that is, it does not transfer responsibility for the object from the service to the client. This is accomplished by means of the *finish-delivery*( ) function.
2. Although the *wait*( ) or *size*( ) function indicated that the delivery queue contained an object immediately prior to invocation of this function, this does not guarantee this function's success. For example, another process might have begun the object's delivery.

ARGUMENTS

Session (Private Object)
: An established MA session between the client and the service; an instance of the Session class.

RESULTS

Return Code (Return Code)
: Whether the function succeeded and, if not, why. It may be **success** or one of the values listed under Errors below.

Object (Private Object)
: The object whose delivery is started; an instance of the Delivered Message or the Delivered Report class. If the former, the object includes one envelope for each of one or more of the users associated with the session. The service prevents the client from modifying this object subsequently. This result is present if, and only if, the Return Code result is success.

ERRORS

bad-message, feature-unavailable, function-interrupted, memory-insufficient, network-error, no-such-session, not-private, permanent-error, pointer-invalid, queue-empty, session-busy, system-error, temporary-error, wrong-class.

NAME

start-retrieval - begin the retrieval of a message or a report

SYNOPSIS

```
#include <xmh.h>

OM_return_code
ma_start_retrieval (
    OM_private_object     session,
    MH_sequence_number    minimum_sequence_number,
    MH_sequence_number   *selected_sequence_number,
    OM_private_object    *object
);
```

DESCRIPTION

This function begins the retrieval of a message or a report by reserving an unreserved object in the retrieval queue to which a session provides access. The client shall finish the retrieval of one object before it starts the retrieval of another in the same session. The retrieval of a particular object cannot simultaneously be in progress in two sessions.

The service selects for retrieval the object whose sequence number is nearest to but no less than a sequence number specified by the client. If no object has a sequence number greater than or equal to that specified, the function reports an exception.

Notes:

1. The invocation of this function initiates, but does not consummate, retrieval. That is, it does not transfer responsibility for the object from the service to the client. That is accomplished, if desired, by means of the function.
2. Although the *wait*( ) or *size*( ) function indicated that the retrieval queue contained an object immediately prior to invocation of this function, this does not guarantee this function's success. For example, another process might have begun the object's retrieval.

ARGUMENTS

Session (Private Object)
An established MA session between the client and the service; an instance of the Session class.

Minimum Sequence Number (Sequence Number)
The sequence number of the first message or report to be considered for retrieval.

RESULTS

Return Code (Return Code)
Whether the function succeeded and, if not, why. It may be **success** or one of the values listed under Errors below.

Selected Sequence Number (Sequence Number)
The sequence number of the message or report selected for retrieval. This result is present if, and only if, the Return Code result is success.

Object (Private Object)
The object whose retrieval is started; an instance of the Delivered Message or the Delivered Report class. If the former, the object includes a single envelope for the user associated with the session. The service prevents the client from modifying this object subsequently. This result is present if, and only if, the Return Code result is success.

ERRORS

feature-unavailable, function-interrupted, memory-insufficient, network-error, no-such-session, not-private, permanent-error, pointer-invalid, queue-empty, session-busy, system-error, temporary-error, wrong-class.

NAME

submit - submit a communique

SYNOPSIS

```
#include <xmh.h>

OM_return_code
ma_submit (
    OM_private_object    session,
    OM_object            communique,
    OM_private_object   *submission_results
);
```

DESCRIPTION

This function submits a communique by adding it to the submission queue to which a session provides access. This transfers responsibility for the communique from the client to the service. The function first verifies the communique's integrity.

ARGUMENTS

Session (Private Object)

An established MA session between the client and the service; an instance of the Session class.

Communique (Object)

The object to be submitted; an instance of the Submitted Communique class. Its purported originator shall be among the users associated with the session. The communique is made inaccessible if it is a private object. If the object is Public, and contains attributes holding **OM_string** values in which the Local String bit of the syntax is set, then the strings are converted according to the rules that apply for the *om_put( )* function.

RESULTS

Return Code (Return Code)

Whether the function succeeded and, if not, why. It may be **success** or one of the values listed under Errors below.

Submission Results (Private Object)

The results of the submission; an instance of the Submission Results class. This result is present if, and only if, the Return Code result is success.

ERRORS

feature-unavailable, function-interrupted, memory-insufficient, network-error, no-such-class, no-such-object, no-such-session, no-such-syntax, no-such-type, not-private, originator-improper, permanent-error, pointer-invalid, system-error, temporary-error, too-many-values, unsupported-critical-function wrong-class, wrong-value-length, wrong-value-makeup, wrong-value-number, wrong-value-syntax, wrong-value-type.

NAME

wait - return when a message or a report is available for delivery or retrieval, or when a period of time elapses, whichever occurs first

SYNOPSIS

```
#include <xmh.h>

OM_return_code
ma_wait (
    OM_private_object     session,
    MH_interval           interval,
    MH_sequence_number    minimum_sequence_number,
    OM_boolean           *available
);
```

DESCRIPTION

This function returns when a message or a report is available for delivery or retrieval in the delivery or retrieval queue to which a session provides access, or when a period of time elapses, whichever occurs first. For retrieval queues, a message or a report is considered to be available when its sequence number is not less than a sequence number specified by the client.

For delivery queues:

- If there is at least one unreserved object on the queue to which the session provides access, the call returns immediately and flags available.
- If there are no unreserved objects on the queue and the specified interval is zero, the call returns immediately and flags unavailable.
- If there are no unreserved objects on the queue and the specified interval is greater than zero, then the call returns after the specified interval has elapsed or when the service places a new object on the queue, whichever occurs first. In the former case, availability is false, in the latter true.
- If there are multiple sessions waiting on a single queue when an object arrives on the queue, to which and to how many the service returns early is implementation- defined.

For retrieval queues:

- If there is at least one unreserved object, with sequence number greater than or equal to a sequence number specified by the client, and this object is on the queue to which the session provides access, then the call returns immediately and flags available.
- If all the unreserved objects on the queue have sequence numbers strictly less than a sequence number specified by the client and the specified interval is zero, then the call returns immediately and flags unavailable.
- If all the unreserved objects on the queue have sequence numbers strictly less than a sequence number specified by the client and the specified interval is greater than zero, then the call returns after the specified interval has elapsed or when the service places a new object on the queue having sequence number greater than or equal to the sequence number specified by the client, whichever occurs first. In the former case, availability is false, in the latter true.
- If there are multiple sessions waiting on a single queue when an object arrives on the queue, to which and to how many the service returns early is implementation-defined.

Note: This function is designed to be easily implemented using the event signalling primitives of many operating systems, including the primitives whose inclusion in POSIX is currently under discussion within IEEE.

ARGUMENTS

Session (Private Object)
An established MA session between the client and the service; an instance of the Session class.

Interval (Interval)
The maximum length of time that the service is to block the client before returning.

Minimum Sequence Number (Sequence Number)
The sequence number, less than which objects on a retrieval queue are considered unavailable. This argument is absent for delivery queues.

In the C interface, this argument is present for delivery queues, but is ignored by the service.

RESULTS

Return Code (Return Code)
Whether the function succeeded and, if not, why. It may be **success** or one of the values listed under Errors below.

Available (Boolean)
True if there was an available object on the queue when the function was called or if the call returned early when an available object was placed on the queue.

ERRORS

feature-unavailable, function-interrupted, memory-insufficient, network-error, no-such-session, not-private, permanent-error, pointer-invalid, system-error, temporary-error, wrong-class.

3.4 Transfer Functions

This section defines, and Table 3-4 lists, the functions of the MT interface. The functions of both the generic and C interfaces are specified. Those of the C interface are repeated in Section 3.6 on page 56, which serves as a summary and a reference.

Function	Description
Close	Terminate an MT session.
Finish Transfer In	Conclude one of the transfers in in progress in a session.
Open	Establish an MT session.
Size	Determine the size of the input queue.
Start Transfer In	Begin the transfer in of a communique or a report.
Transfer Out	Add a communique or report to the output queue.
Wait	Return when an object is available for transfer in.

Table 3-4 MT Interface Functions

As indicated in the table, the MT interface comprises a number of functions whose purpose and range of capabilities are summarised as follows:

Close
This function terminates an MT session between the client and the service. If the transfer in of a communique or a report is in progress, the service first unsuccessfully finishes that transfer in.

Finish Transfer In
This function concludes one or all of the transfers in in progress in a session. The client indicates whether the service is to remove the objects from the input queue, or leave them there.

Open
This function establishes an MT session between the client and the service, and makes the Basic Transfer FU and the OM Package initially available in that session. The client may also specify the other features required for the session. The client specifies its name and instance name.

Size
This function determines the number of unreserved objects in the input queue to which a session provides access. Each object is a communique or a report.

Start Transfer In
This function begins the transfer in of a communique or a report by reserving an unreserved object in the input queue. If no objects await transfer in, the function reports an exception.

Transfer Out
This function adds a communique or a report to the output queue to which a session provides access. The function first verifies the object's integrity.

Wait
This function returns when a communique or a report is available for transfer in the input queue to which a session provides access, or when a period of time elapses, whichever occurs first.

The functions are grouped into three FUs (one basic and one each for transfer in and transfer out) as indicated in Table 3-5.

Basic Transfer	Transfer Out	Transfer In
Open Close OM API	Transfer Out	Size Start Transfer In Finish Transfer In Wait

Table 3-5 MT Interface Functional Units

Note: The OM API is defined in the referenced **XOM** Specification.

The intent of the interface definition is that each function is atomic, that is, it either carries out its assigned task in full and reports success, or fails to carry out even a portion of the task and reports an exception. However, the service does not guarantee that a task will not occasionally be carried out in part but not in full.

Note: Making such a guarantee might be prohibitively expensive.

Whether a function detects and reports each of the exceptions listed in the Errors clause of its specification is unspecified. If a function detects two or more exceptions, which it reports is unspecified. If a function reports an exception for which a return code is defined, however, it uses that (rather than another) return code to do so.

NAME

close - terminate an MT session between the client and the service

SYNOPSIS

```
#include <xmh.h>

OM_return_code
mt_close (
    OM_private_object    session
);
```

DESCRIPTION

This function terminates an MT session, as well as destroys the workspace, between the client and the service. If the transfer in of one or more communiques or reports are in progress in the session, the service first unsuccessfully finishes those transfers in.

ARGUMENTS

Session (Private Object)

An established MT session between the client and the service; an instance of the Session class.

RESULTS

Return Code (Return Code)

Whether the function succeeded and, if not, why. It may be **success** or one of the values listed under Errors below.

ERRORS

function-interrupted, memory-insufficient, network-error, no-such-session, not-private, permanent-error, pointer-invalid, system-error, temporary-error, wrong-class.

NAME

finish-transfer-in - conclude one or all of the transfers in progress in a session

SYNOPSIS

```
#include <xmh.h>

OM_return_code
mt_finish_transfer_in (
    OM_private_object    session,
    OM_private_object    object,
    OM_boolean           remove
);
```

DESCRIPTION

This function concludes one or all of the transfers in progress in a session. The client indicates whether the service is to remove the objects from the input queue or leave them there.

In the former case, the service considers that it has transferred responsibility for the object from the service to the client; the object handle remains accessible on return from this function and may be deleted (using the OM Delete function) when no longer required.

In the latter case, the object is returned to the retrieval queue and marked unreserved. The associated object handle is made inaccessible on return from this function and should not be deleted by the client; however, the associated communique or report can be obtained again by a subsequent *start-retrieval*( ) function invocation.

The client should not invoke the OM Delete function on the object prior to calling this function; otherwise the behaviour of this function is undefined.

Notes:

1. The client should invoke this function, with the Remove argument true, after it successfully processes an object in the input queue. If the object is a message or a report, such processing involves delivering or internally transferring the object to the indicated users of the mail system to which the client is a gateway. If the object is a probe, such processing involves establishing the deliverability to those users of a message of the variety the probe describes.
2. An exceptional condition may prevent the client from successfully processing the object. If it judges the exception (for example, the invalidity of an O/R address) to be permanent, the client should issue an NDR and permanently terminate the transfer in using the present function, with the Remove argument true. (However, X.400 does not permit an undeliverable report itself to provoke an NDR, but rather requires that the report simply be discarded.) If the client judges the exception (for example, a network failure) to be temporary, the client should temporarily terminate the transfer in using the present function, with the Remove argument false, and later reinitiate it using the *start-transfer-in*( ) function.

ARGUMENTS

Session (Private Object)
: An established MT session between the client and the service; an instance of the Session class.

Object (Private Object)
: The particular object whose transfer in is to be concluded; an instance of the Communique or the Report class. The object shall be (currently) reserved. This argument is present if, and only if, the All argument is false.

All (Boolean)
Whether the transfer in of all reserved objects, rather than one particular reserved object, is to be concluded.

In the C interface, this argument is true if, and only if, the Object argument is the null pointer.

Remove (Boolean)
Whether the service is to remove the object or objects from the input queue, rather than leave them there.

RESULTS

Return Code (Return Code)
Whether the function succeeded and, if not, why. It may be **success** or one of the values listed under Errors below.

ERRORS

feature-unavailable, function-interrupted, memory-insufficient, network-error, no-such-object, no-such-session, not-private, not-reserved, permanent-error, pointer-invalid, session-not-busy, system-error, temporary-error, wrong-class.

NAME

open - establish an MT session between the client and the service

SYNOPSIS

```
#include <xmh.h>

OM_return_code
mt_open (
    OM_string            client_name,
    OM_string            client_instance_name,
    MH_feature           feature_list[],
    OM_private_object    *session,
    OM_workspace         *workspace
);
```

DESCRIPTION

This function establishes an MT session between the client and the service, and makes the Basic Transfer FU and the OM Package initially available in that session. The client may also specify the other features required for the session.

The client specifies its own name and its instance name. If the service has several clients, the client's name determines the input and output queues to which the session provides access. All *client instances* share the client's input and output queues. If the client has several instances, the service may use the instance name, for example, in log entries it makes for purposes of system management.

Opening an MT session also creates a workspace. A workspace contains objects returned as a result of functions invoked within that session. The workspace is used as an argument in the *om_create*( ) and *copy*( ) functions.

The maximum number of sessions that may exist simultaneously is implementation-defined and may vary with time.

ARGUMENTS

Client Name (String)
: The name by which the service knows the client, interpreted as a value whose syntax is String (IA5).

Client Instance Name (String)
: The name by which the service knows the client instance, interpreted as a value whose syntax is String (IA5).

Feature-List (Feature-List)
: An ordered sequence of features, each represented by an object identifier. The sequence is terminated by an object identifier having no components (a length of zero and any value of the data pointer in the C representation).

RESULTS

Return Code (Return Code)
: Whether the function succeeded and, if not, why. It may be **success** or one of the values listed under Errors below.

Activated (Boolean-List)

If the function completed successfully, this result contains an ordered sequence of Boolean values, with the same number of elements as the Feature-List. If true, each value indicates that the corresponding feature is now part of the interface. If false, each value indicates that the corresponding feature is not available.

In the C binding, this result is combined with the Feature-List argument as a single array of structures of type **MH_feature**.

Session (Private Object)

The established MT session between the client and the service; an instance of the Session class. The service prevents the client from modifying this object subsequently. This result is present if, and only if, the Return Code result is success.

Workspace (OM_workspace)

The workspace that will contain all objects returned as a result of the functions invoked in the session.

ERRORS

feature-conflicts, feature-unavailable, function-interrupted, memory-insufficient, network-error, no-such-client, no-such-client-instance, permanent-error, pointer-invalid, system-error, temporary-error, too-many-sessions, wrong-class.

NAME

size - determine the number of communiques or reports in the input queue

SYNOPSIS

```
#include <xmh.h>

OM_return_code
mt_size (
    OM_private_object    session,
    MH_object_count     *number
);
```

DESCRIPTION

This function determines the number of communiques and reports in the input queue to which a session provides access.

ARGUMENTS

Session (Private Object)
: An established MT session between the client and the service; an instance of the Session class.

RESULTS

Return Code (Return Code)
: Whether the function succeeded and, if not, why. It may be **success** or one of the values listed under Errors below.

Number (Object Count)
: The number of unreserved objects in the input queue. However, if that number exceeds 2^{16}-1, 2^{16}-1 is returned. This result is present if, and only if, the Return Code result is **success**.

ERRORS

feature-unavailable, function-interrupted, memory-insufficient, network-error, no-such-session, not-private, permanent-error, pointer-invalid, system-error, temporary-error, wrong-class.

NAME

Start Transfer In - begin the transfer in of a communique or a report

SYNOPSIS

```
#include <xmh.h>

OM_return_code
mt_start_transfer_in (
    OM_private_object     session,
    OM_private_object    *object,
    OM_string            *identifier
);
```

DESCRIPTION

This function begins the transfer in of a communique or a report by reserving an unreserved object in the input queue to which a session provides access. If no objects await transfer in, the function reports an exception. The client need not finish the transfer in of one object before it starts the transfer in of another in the same session. The transfer in of a particular object, however, cannot simultaneously be in progress in two sessions.

Whether the service begins the transfer in of a message addressed to several users in one function invocation or several (for example, one per user or per group of users) is implementation-defined.

Which qualifying object in the input queue (if there are several such objects) the service selects for transfer in is implementation-defined.

Notes:

1. The invocation of this function initiates, but does not consummate, transfer in, that is, it does not transfer responsibility for the object from the service to the client. That is accomplished by means of the *finish-transfer-in*( ) function.
2. Although the *wait*( ) or *size*( ) function indicated that the input queue contained an object immediately prior to invocation of this function, this does not guarantee this function's success. For example, another process might have begun the object's transfer-in.

ARGUMENTS

Session (Private Object)
: An established MT session between the client and the service; an instance of the Session class.

RESULTS

Return Code (Return Code)
: Whether the function succeeded and, if not, why. It may be **success** or one of the values listed under Errors below.

Object (Private Object)
: The object whose transfer in is started; an instance of the Communique or the Report class. The service prevents the client from modifying this object subsequently. This result is present if, and only if, the Return Code result is success.

Identifier (String)

A service-assigned identifier that distinguishes the object from all other objects ever placed, or ever to be placed (for all time) in the input queue. The identifier is interpreted as a value whose syntax is `String` (IA5). Its maximum length is 64 characters. This result is present if, and only if, the Return Code result is **success**.

In the C interface, the client provides, by means of a pointer, the storage into which the service is to place the returned identifier and its actual length. That space must be sufficient to accommodate the longest allowed identifier. It is also the client's responsibility to deallocate the storage whenever the identifier is no longer useful.

ERRORS

bad-message, feature-unavailable, function-interrupted, memory-insufficient, network-error, no-such-session, not-private, permanent-error, pointer-invalid, queue-empty, session-busy, system-error, temporary-error, wrong-class.

NAME

transfer-out - add a communique or a report to the output queue

SYNOPSIS

```
#include <xmh.h>

OM_return_code
mt_transfer_out (
    OM_private_object    session,
    OM_object            object
);
```

DESCRIPTION

This function adds a communique or a report to the output queue to which a session provides access. This transfers responsibility for that object from the client to the service. The function first verifies the object's integrity.

ARGUMENTS

Session (Private Object)

An established MT session between the client and the service; an instance of the Session class.

Object (Object)

The object to be added to the output queue; an instance of the Communique or the Report class. It shall not be reserved in order to avoid the object being made inaccessible before calling *mt_finish_transfer_in*( ) for it. If the object is private, it is made inaccessible. If the object is Public, and contains attributes holding *OM_string* values in which the Local String bit of the syntax is set, then the strings are converted according to the rules that apply for the *om_put*( ) function.

RESULTS

Return Code (Return Code)

Whether the function succeeded and, if not, why. It may be **success** or one of the values listed under Errors below.

ERRORS

feature-unavailable, function-interrupted, memory-insufficient, network-error, no-such-class, no-such-object, no-such-session, no-such-syntax, no-such-type, not-private, not-unreserved, permanent-error, pointer-invalid, system-error, temporary-error, too-many-values, wrong-class, wrong-value-length, wrong-value-makeup, wrong-value-number, wrong-value-syntax, wrong-value-type.

NAME

wait - return when a communique or a report is available for transfer in, or when a period of time elapses, whichever occurs first

SYNOPSIS

```
#include <xmh.h>

OM_return_code
mt_wait (
    OM_private_object    session,
    MH_interval          interval,
    OM_boolean          *available
);
```

DESCRIPTION

This function returns when a communique or a report is available for transfer in in the input queue to which a session provides access, or when a period of time elapses, whichever occurs first.

If there is at least one unreserved object on the input queue to which the session provides access, the call returns immediately and flags available.

If there are no unreserved objects on the input queue and the specified interval is zero, the call returns immediately and flags unavailable.

If there are no unreserved objects on the input queue and the specified interval is greater than zero, then the call returns after the specified interval has elapsed or when the service places a new object on the queue, whichever occurs first. In the former case availability is false, in the latter true.

If there are multiple sessions waiting on a single queue when an object arrives on the queue, to which and to how many the service returns early is implementation- defined.

Note: This function is designed to be easily implemented using the event signalling primitives of many operating systems, including the primitives whose inclusion in POSIX is currently under discussion within IEEE.

ARGUMENTS

Session (Private Object)
An established MT session between the client and the service; an instance of the Session class.

Interval (Interval)
The maximum length of time that the service is to block the client before returning.

RESULTS

Return Code (Return Code)
Whether the function succeeded and, if not, why. It may be **success** or one of the values listed under Errors below.

Available (Boolean)
True if there was an available object on the queue when the function was called or if the call returned early when an object was placed on the queue.

ERRORS

feature-unavailable, function-interrupted, memory-insufficient, network-error, no-such-session, not-private, permanent-error, pointer-invalid, system-error, temporary-error, wrong-class.

3.5 Return Codes

This section defines, and the following Table 3-6 and Table 3-7 list, the return codes of the MA and MT interfaces that are specific to MH. The return codes of the generic interface alone are specified here. The return codes of the C interface are specified in the declaration summary in Section 3.6 on page 56. The interfaces also include certain return codes of the OM interface. These too are included in the tables below, where their names appear in italics.

The tables' first column lists the return codes. The other columns identify with an ''x'' the return codes that apply to each function. (This information, organised differently, also appears in the Errors clauses of the function descriptions for the Access Functions (Section 3.3 on page 24) and Transfer Functions (Section 3.4 on page 41).

Return Code	**Can**	**Clo**	**FiD**	**FiR**	**Ope**	**Siz**	**StD**	**StR**	**Sub**	**Wai**
bad-message	-	-	-	-	-	-	x	-	-	-
feature-conflicts	-	-	-	-	x	-	-	-	-	-
feature-unavailable	x	-	x	x	x	x	x	x	x	x
function-interrupted	x	x	x	x	x	x	x	x	x	x
memory-insufficient	x	x	x	x	x	x	x	x	x	x
network-error	x	x	x	x	x	x	x	x	x	x
no-such-class	-	-	-	-	-	-	-	-	x	-
no-such-client	-	-	-	-	x	-	-	-	-	-
no-such-client-instance	-	-	-	-	-	-	-	-	-	-
no-such-message	x	-	-	-	-	-	-	-	-	-
no-such-object	-	-	x	x	-	-	-	-	x	-
no-such-session	x	x	x	x	-	x	x	x	x	x
no-such-syntax	x	-	x	-	x	-	-	-	x	-
no-such-type	x	-	x	-	x	-	-	-	x	-
no-such-user	-	-	-	-	x	-	-	-	-	-
not-private	x	x	x	x	-	x	x	x	x	x
not-reserved	-	-	-	-	-	-	-	-	-	-
not-unreserved	-	-	-	-	-	-	-	-	-	-
originator-improper	-	-	-	-	-	-	-	-	x	-
permanent-error	x	x	x	x	x	x	x	x	x	x
pointer-invalid	x	x	x	x	x	x	x	x	x	x
queue-empty	-	-	-	-	-	-	x	x	-	-
session-busy	-	-	-	-	-	-	x	x	-	-
session-not-busy	-	-	x	x	-	-	-	-	-	-
success	x	x	x	x	x	x	x	x	x	x
system-error	x	x	x	x	x	x	x	x	x	x
temporary-error	x	x	x	x	x	x	x	x	x	x
too-late	x	-	-	-	-	-	-	-	-	-
too-many-sessions	-	-	-	-	x	-	-	-	-	-
too-many-values	x	-	x	-	x	-	-	-	x	-
unsupported-critical-function	-	-	-	-	-	-	-	-	x	-
wrong-class	x	x	x	x	x	x	x	x	x	x
wrong-value-length	x	-	x	-	x	-	-	-	x	-
wrong-value-makeup	x	-	x	-	x	-	-	-	x	-
wrong-value-number	x	-	x	-	x	-	-	-	x	-
wrong-value-syntax	x	-	x	-	x	-	-	-	x	-
wrong-value-type	x	-	x	-	x	-	-	-	x	-

Table 3-6 MA Interface Return Codes

Return Code	Clo	FiT	Ope	Siz	StT	TrO	Wai
bad-message	-	-	-	-	x	-	-
feature-conflicts	-	-	x	-	-	-	-
feature-unavailable	-	x	x	x	x	x	x
function-interrupted	x	x	x	x	x	x	x
memory-insufficient	x	x	x	x	x	x	x
network-error	x	x	x	x	x	x	x
no-such-class	-	-	-	-	-	x	-
no-such-client	-	-	x	-	-	-	-
no-such-client-instance	-	-	x	-	-	-	-
no-such-message	-	-	-	-	-	-	-
no-such-object	-	x	-	-	-	x	-
no-such-session	x	x	-	x	x	x	x
no-such-syntax	-	-	-	-	-	x	-
no-such-type	-	-	-	-	-	x	-
no-such-user	-	-	-	-	-	-	-
not-private	x	x	-	x	x	x	x
not-reserved	-	x	-	-	-	-	-
not-unreserved	-	-	-	-	-	x	-
originator-improper	-	-	-	-	-	-	-
permanent-error	x	x	x	x	x	x	x
pointer-invalid	x	x	x	x	x	x	x
queue-empty	-	-	-	-	x	-	-
session-busy	-	-	-	-	x	-	-
session-not-busy	-	x	-	-	-	-	-
success	x	x	x	x	x	x	x
system-error	x	x	x	x	x	x	x
temporary-error	x	x	x	x	x	x	x
too-late	-	-	-	-	-	-	-
too-many-sessions	-	-	x	-	-	-	-
too-many-values	-	-	-	-	-	x	-
wrong-class	x	x	x	x	x	x	x
wrong-value-length	-	-	-	-	-	x	-
wrong-value-makeup	-	-	-	-	-	x	-
wrong-value-number	-	-	-	-	-	x	-
wrong-value-syntax	-	-	-	-	-	x	-
wrong-value-type	-	-	-	-	-	x	-

Table 3-7 MT Interface Return Codes

The MH-specific return codes are as follows:

bad-message
The first available message (or report) in the queue is badly formed.

feature-conflicts
The requested feature conflicts with an existing feature of the session.

feature-unavailable
The service does not offer the requested feature.

no-such-client
The service does not recognise the specified client name.

no-such-client-instance
The service does not recognise the specified client instance name.

no-such-message
A message's deferred delivery cannot be cancelled because the MTS identifier does not identify a message previously submitted by the client.

no-such-session
The specified session does not exist.

no-such-user
The service does not recognise the specified O/R address.

not-reserved
The specified object is not reserved.

not-unreserved
The specified object is reserved.

originator-improper
The user identified as the originator of a submitted communique is not among those associated with the session.

queue-empty
No objects are available for delivery, retrieval, or transfer in in the delivery, retrieval, or input queue to which the session provides access.

session-busy
A delivery or retrieval, or the maximum number of transfers in, are already in progress in the session.

session-not-busy
A delivery, retrieval, or transfer in is not in progress in the session.

too-late
A message's deferred delivery cannot be cancelled because either the message has been progressed for transfer or delivery, or the service has provided the client with proof of submission.

too-many-sessions
An implementation-defined limitation prevents the establishment of another session.

unsupported-critical-function
The client has specified an argument of the operation, marked as critical for submission, but unsupported by the service.

wrong-class
An object is an instance of the wrong class.

3.6 Declaration Summary

This section lists the declarations that define the C language MA and MT interfaces. All of the declarations, except those for symbolic constants, also appear in the specifications for Data Types (Section 3.2 on page 22), Access Functions (Section 3.3 on page 24) or Transfer Functions (Section 3.4 on page 41).

The declarations, as assembled here, constitute the contents of a header file to be made accessible to client programmers. The header file is **<xmh.h>**. The symbols the declarations define are the only symbols the service makes visible to the client, with the exception of the symbols that define the various packages.

```
/* BEGIN MA AND MT INTERFACES */

/* DATA TYPES */

/* Feature List */

typedef struct {
    OM_object_identifier    feature,
    OM_boolean              activated,
} MH_feature;

/* Interval */

typedef OM_uint32 MH_interval;

/* Object Count */

typedef OM_uint32 MH_object_count;

/* Sequence Number */

typedef OM_uint32 MH_sequence_number;

/* MA FUNCTIONS */

/* Cancel Submission */

OM_return_code
ma_cancel_submission (
    OM_private_object    session,
    OM_object            mts_identifier,
);

/* Close */

OM_return_code
ma_close (
    OM_private_object    session
);

/* Finish Delivery */

OM_return_code
ma_finish_delivery (
    OM_private_object    session,
```

```
    OM_object            delivery_confirmations,
    OM_object            non_delivery_reports,
);

/* Finish Retrieval */

OM_return_code
ma_finish_retrieval (
    OM_private_object    session,
    OM_boolean           remove,
);

/* Open */

OM_return_code
ma_open (
    OM_public_object     user_address,
    OM_string            client_name,
    MH_feature           feature_list[],
    OM_private_object   *session,
    OM_workspace        *workspace,
);

/* Size */

OM_return_code
ma_size (
    OM_private_object    session,
    MH_object_count     *number,
);

/* Start Delivery */

OM_return_code
ma_start_delivery (
    OM_private_object    session,
    OM_private_object   *object,
);

/* Start Retrieval */

OM_return_code
ma_start_retrieval (
    OM_private_object    session,
    MH_sequence_number   minimum_sequence_number,
    MH_sequence_number  *selected_sequence_number,
    OM_private_object   *object,
);

/* Submit */

OM_return_code
ma_submit (
    OM_private_object    session,
    OM_object            communique,
    OM_private_object   *submission_results,
);

/* Wait */
```

```
OM_return_code
ma_wait (
    OM_private_object    session,
    MH_interval          interval,
    MH_sequence_number   minimum_sequence_number,
    OM_boolean          *available,
);

/* MT FUNCTIONS */

/* Close */

OM_return_code
mt_close (
    OM_private_object    session,
);

/* Finish Transfer In */

OM_return_code
mt_finish_transfer_in (
    OM_private_object    session,
    OM_private_object    object,
    OM_boolean           remove,
);

/* Open */

OM_return_code
mt_open (
    OM_string            client_name,
    OM_string            client_instance_name,
    MH_feature           feature_list[],
    OM_private_object   *session,
    OM_workspace        *workspace
);
/* Size */

OM_return_code
mt_size (
    OM_private_object    session,
    MH_object_count     *number
);

/* Start Transfer In */

OM_return_code
mt_start_transfer_in (
    OM_private_object    session,
    OM_private_object   *object,
    OM_string           *identifier
);

/* Transfer out */

OM_return_code
mt_transfer_out (
    OM_private_object    session,
    OM_object            object
```

```
);

/* Wait */

OM_return_code
mt_wait (
    OM_private_object    session,
    MH_interval          interval,
    OM_boolean          *available
);

/* SYMBOLIC CONSTANTS */

/* Object Identifiers (Elements component) */


/* Feature */
#define OMP_O_MH_FE_BASIC_ACCESS          "\x2A\x86\x48\xCE\x22\x01"
#define OMP_O_MH_FE_SUBMISSION            "\x2A\x86\x48\xCE\x22\x02"
#define OMP_O_MH_FE_DELIVERY              "\x2A\x86\x48\xCE\x22\x03"
#define OMP_O_MH_FE_RETRIEVAL             "\x2A\x86\x48\xCE\x22\x04"
#define OMP_O_MH_FE_BASIC_TRANSFER        "\x2A\x86\x48\xCE\x22\x05"
#define OMP_O_MH_FE_TRANSFER_IN           "\x2A\x86\x48\xCE\x22\x06"
#define OMP_O_MH_FE_TRANSFER_OUT          "\x2A\x86\x48\xCE\x22\x07"
#define OMP_O_MH_FE_IM_84                 "\x2A\x86\x48\xCE\x22\x08"
#define OMP_O_MH_FE_IM_88                 "\x2A\x86\x48\xCE\x22\x09"
#define OMP_O_MH_FE_MH_84                 "\x2A\x86\x48\xCE\x22\x0A"
#define OMP_O_MH_FE_MH_88                 "\x2A\x86\x48\xCE\x22\x0B"
#define OMP_O_MH_FE_SM_88                 "\x2A\x86\x48\xCE\x22\x0C"
#define OMP_O_MH_FE_MULTIPLE_DELIVERY     "\x2A\x86\x48\xCE\x22\x0D"
#define  OMP_O_MH_FE_GENERAL_CONTENT      "\x2A\x86\x48\xCE\x22\x0E"

/* Return Codes */

#define MH_RC_BAD_MESSAGE                  ( (OM_return_code) 100 )
#define MH_RC_FEATURE_CONFLICTS            ( (OM_return_code) 101 )
#define MH_RC_FEATURE_UNAVAILABLE          ( (OM_return_code) 102 )
#define MH_RC_NO_SUCH_CLIENT               ( (OM_return_code) 103 )
#define MH_RC_NO_SUCH_CLIENT_INST ANCE     ( (OM_return_code) 104 )
#define MH_RC_NO_SUCH_MESSAGE              ( (OM_return_code) 105 )
#define MH_RC_NO_SUCH_USER                 ( (OM_return_code) 106 )
#define MH_RC_NOT_RESERVED                 ( (OM_return_code) 107 )
#define MH_RC_NOT_UNRESERVED               ( (OM_return_code) 108 )
#define MH_RC_ORIGINATOR_IMPROPER          ( (OM_return_code) 109 )
#define MH_RC_QUEUE_EMPTY                  ( (OM_return_code) 110 )
#define MH_RC_SESSION_BUSY                 ( (OM_return_code) 111 )
#define MH_RC_SESSION_NOT_BUSY             ( (OM_return_code) 112 )
#define MH_RC_TOO_LATE                     ( (OM_return_code) 113 )
#define MH_RC_TOO_MANY_SESSIONS            ( (OM_return_code) 114 )
#define MH_RC_UNSUPP_CRITICAL_FUNCTION     ( (OM_return_code) 115 )
#define MH_RC_WRONG_CLASS                  ( (OM_return_code) 116 )
#define MH_RC_NO_SUCH_SESSION              ( (OM_return_code) 117 )

/* END MA AND MT INTERFACES */
```

Chapter 4

Interpersonal Messaging Packages

4.1 Summary

This chapter defines the IM 84 Package and the IM 88 Package. The first provides the functionality of IM (1984), the second that of IM (1988). In broad terms, the latter extends the functionality of the former. The entire chapter applies to both packages, except those aspects marked by the phrase *for 1988 alone*, which apply to the IM 88 Package only. The chapter is to be understood in the context provided by the referenced **XOM** Specification.

Throughout this chapter, the words *originator* and *recipient* refer to the roles that various users play in the conveyance of interpersonal messages (IPMs) and interpersonal notifications (IPNs) via the MTS.

Notes:

1. An IPM may appear in the Body attribute of another IPM which itself is conveyed as the content of a message. The words *originator* and *recipient* are to be understood in the context of an IPM's conveyance as the (entire) content of a message, not as a component of the Body attribute of another IPM so conveyed.
2. X.400's Encrypted, SFD, Telex and Voice body part types are not supported. X.400 (1984) does not fully specify the Encrypted, Telex and Voice types. X.400 (1988) abandons the SFD and Telex types, and leaves the Encrypted and Voice types only partially specified.

4.2 Class Hierarchy

This section depicts the hierarchical organisation of the IM classes. Subclassification is indicated by indentation. The names of abstract classes are in italics. Thus, for example, **Receipt Notification** is an immediate subclass of *Interpersonal Notification*, an abstract class. The names of classes to which the *om_encode*( ) function applies are in bold. The *om_create*( ) function applies to all concrete classes.

Object (defined in the referenced **XOM** Specification)

- *Body Part*
  - — **Bilaterally Defined Body Part**
  - — **Externally Defined Body Part**
  - — **G3 Fax Body Part**
  - — **G4 Class 1 Body Part**
  - — **General Text Body Part**
  - — **IA5 Text Body Part**
  - — **ISO 6937 Text Body Part**
  - — **Message Body Part**
  - — **Mixed-mode Body Part**
  - — **Nationally Defined Body Part**
  - — **Office Document Architecture Body Part**
  - — **Teletex Body Part**
  - — **Unidentified Body Part**
  - — **USA Nationally Defined Body Part**
  - — **Videotex Body Part**
- *Content* (defined in Chapter 5)
  - — **Interpersonal Message**[1]
  - — *Interpersonal Notification*
    - — **Non-receipt Notification**[1]
    - — **Receipt Notification**[1]
- **IPM Identifier**
- **OR Descriptor**
- **Recipient Specifier**

[1] These classes are encodable if the SM 88 Package has been negotiated for the session. The encoding shall be of the `Information Object` syntax as specfied in X420.

Note: The identifier for the variable name of type OM_STRING of a class in the Interpersonal Messaging package can usually be derived using the name of the class, preceded by "IM_C_", and replacing a blank space with an underscore. To be in line with the ANSI C language limitation, some words in the class names are excepted and are abbreviated as below:

NOTIFICATION	is abbreviated as	NOTIF
BODY_PART		BD_PRT
BILATERALLY		BILAT
EXTERNALLY		EXTERN
NATIONALLY		NATIONAL

4.3 Class Definitions

This section defines the IM classes. It describes the attributes specific to a class in a table like those used in the referenced **XOM** Specification. The table includes, however, an additional column that identifies the attributes that are for 1988 alone.

A class, all the attributes specific to which are for 1988 alone, is called a *1988 class*. A 1988 class is not in the IM 84 Package, although it may be in its closure.

Note: The value length restrictions were introduced to X.400 by means of the **X.400-Series Implementors' Guide** (see Referenced Documents).

4.3.1 Bilaterally Defined Body Part

An instance of class **Bilaterally Defined Body Part** comprises arbitrary binary data. The instance itself indicates neither the syntax nor the semantics of that data. Rather, the client is presumed to know these a priori.

The attributes specific to this class are listed in the following table.

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
Bilateral Data	String (Octet)	-	1	-	-

Table 4-1 Attributes Specific to Bilaterally Defined Body Part

Bilateral Data
The binary data.

Note: This class was added to X.400 (1984) by means of the **X.400-Series Implementors' Guide** (see Referenced Documents).

4.3.2 Body Part

An instance of class *Body Part* is one of perhaps several information objects (for example, documents) which the IPM containing the instance serves to convey from the IPM's originator to its recipients.

The attributes specific to this class are none.

Note: Under some circumstances, the MTS may subject an IPM to conversion while transferring it between users. Such an event may alter a body part's class, that is, change it from one immediate subclass of Body Part to another.

4.3.3 Externally Defined Body Part

An instance of class **Externally Defined Body Part** comprises an information object whose syntax and semantics are not defined by this document. This class should only be used where the information object cannot be conveyed using a body part class defined elsewhere in this document.

The attributes specific to this class are listed in the following table.

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
External Data	Object (External[1])	-	1	-	1988
External Parameters	Object (External[1])	-	0-1	-	1988

[1]As defined in the **XOM Specification** (see Referenced Documents).

Table 4-2 Attributes Specific to Externally Defined Body Part

External Data
The information object that the body part is intended to convey. Its Direct Reference attribute shall be present, its Data Value Descriptor and Indirect Reference attributes shall be absent.

External Parameters
If present, an information object that characterises the value of the External Data attribute. Its Direct Reference attribute shall be present and allocated by the naming authority that allocated the like-named attribute of the External Data attribute, and at the same time. Its Data Value Descriptor and Indirect Reference attributes shall be absent.

Notes:

1. The MTS may subject an instance of this class to conversion. However, specification of the conversion algorithms may be outside the scope of **X.408** (see Referenced Documents).
2. This class enables the exchange of information objects of all kinds, each unambiguously and uniquely identified. This identification relies upon the Direct Reference attribute mentioned above, which is an object identifier. Object identifiers are easily obtained, for example, by national bodies and private organisations.

4.3.4 G3 Fax Body Part

An instance of class **G3 Fax Body Part** comprises G3 facsimile images (that is, pages).

The attributes specific to this class are listed in the following table.

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
G3 Fax NBPs	Object (G3 Fax NBPs)	-	0-1	-	-
Images	String (Bit)	-	0 or more	-	-

Table 4-3 Attributes Specific to G3 Fax Body Part

G3 Fax NBPs
The NBPs of the images. This attribute is present if (but not only if) the IPM contains two or more G3 fax body parts. If the IPM comprises only a single G3 fax body part, its NBPs may (but need not) be conveyed instead by means of the EITs attribute of the message containing the IPM.

Images
The G3 facsimile images, one per attribute value.

4.3.5 G4 Class 1 Body Part

An instance of class **G4 Class 1 Body Part** comprises a final-form document of the kind that Group 4 (G4) Class 1 facsimile terminals can process.

The attributes specific to this class are listed in the following table.

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
G4 Class 1 Document	String (Octet)	-	0 or more	-	-

Table 4-4 Attributes Specific to G4 Class 1 Body Part

G4 Class 1 Document
The final-form document. Each value of this attribute is a protocol element that contributes to a description of the document's layout structure. Each value shall follow the rules for G4 Class 1 facsimile, which include the BER.

4.3.6 General Text Body Part

An instance of class General Text Body Part comprises General Text.

The attributes specific to this class are listed in the following table.

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
Char Set Registration	Integer	-	0 or more	-	1988
General Text Data	String (General)	-	1	-	1988

Table 4-5 Attributes Specific to General Text Body Part

Char Set Registration
This is the list of the character set registrations that are or may be present in the data component. External EITs are defined for this body part. One EIT is used for each character set identified in this list.

General Text Data
The text. Lines may be of any length. Whenever the data is rendered, all of the text must be communicated.

4.3.7 IA5 Text Body Part

An instance of class **IA5 Text Body Part** comprises IA5 text.

The attributes specific to this class are listed in the following table.

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
Repertoire	Enum (IA5 Repertoire)	-	1	IA5	-
Text	String (IA5)	-	1	-	-

Table 4-6 Attributes Specific to IA5 Text Body Part

Repertoire

Identifies the character set to which the text is constrained. For its defined values, see Section 4.4.3 on page 78.

Text

The text. It may contain lines of any length. Whenever the text is rendered (for example, displayed to or printed for a user), all (not only a part) of the text shall be communicated (for example, lines may be folded but shall not be truncated).

Notes:

1. Within the text, the end of a line is denoted, in common practice, by a carriage return followed by a line feed.
2. Many terminals have a maximum line length of 80 characters. Therefore, lines that do not exceed that length are most likely to be satisfactorily rendered (for example, are most likely to avoid being folded).

4.3.8 Interpersonal Message

An instance of class **Interpersonal Message** (IPM) is a primary information object conveyed between users in IM.

Note: An IPM may be likened to a business memo. In fact, X.400 uses the terms *heading* and *body* in an appeal to that analogy. In X.400 the term *heading* denotes all attributes of an IPM except its Body attribute. Thus the heading of an IPM gives various characteristics of the IPM (for example, its importance) while the body comprises the information objects (for example, documents) that the IPM is intended to convey between users.

The attributes specific to this class are listed in the following table.

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
Authorising Users	Object (OR Descriptor)	-	0 or more	-	-
Auto-forwarded	Boolean	-	1	false	-
Blind Copy Recipients	Object (Recipient Specifier)	-	0 or more	-	-
Body	Object (Body Part)	-	0 or more	-	-
Copy Recipients	Object (Recipient Specifier)	-	0 or more	-	-
Expiry Time	String (UTC Time)	0-17	0-1	-	-
Importance	Enum (Importance)	-	1	routine	-
Incomplete Copy	Boolean	-	1	false	1988
Languages	String (Printable)	2	0 or more	-	1988
Obsoleted IPMs	Object (IPM Identifier)	-	0 or more	-	-
Originator	Object (OR Descriptor)	-	0-1	-	-
Primary Recipients	Object (Recipient Specifier)	-	0 or more	-	-
Related IPMs	Object (IPM Identifier)	-	0 or more	-	-
Replied-to IPM	Object (IPM Identifier)	-	0-1	-	-
Reply Recipients	Object (OR Descriptor)	-	0 or more	-	-
Reply Time	String (UTC Time)	0-17	0-1	-	-
Sensitivity	Enum (Sensitivity)	-	1	not-sensitive	-
Subject	String (Teletex[1])	0-128[2]	0-1	-	-
This IPM	Object (IPM Identifier)	-	1	-	-

[1]Restricted to the graphic subset of the indicated character set.

[2]For 1988 alone, a length of zero is discouraged.

Table 4-7 Attributes Specific to Interpersonal Message

Authorising Users
O/R descriptors for the authorising users. An authorising user is one who, either individually or in concert with others, authorises the origination of an IPM. The word ''authorises'' is not precisely defined by this document, but rather is given meaning by users. This attribute is present if, and only if, the authorising users are other than the IPM's originator alone.

Note: If, for example, a manager instructs his or her secretary to originate an IPM, the secretary, the IPM's originator, might consider the manager the authorising user.

Auto-forwarded
Whether the IPM is the result of automatic (not manual) forwarding.

Blind Copy Recipients
Recipient specifiers for blind copy recipients of the IPM. A blind copy recipient is a copy recipient whose role as such is disclosed to neither primary nor copy recipients. In the IPM (instance) intended for a blind copy recipient, this attribute shall have a value identifying

that user. Whether it has other values that identify the other blind copy recipients is a local matter. In the IPM (instance) intended for a primary or copy recipient, the attribute shall be absent.

Body
The information objects (for example, documents) that the IPM is intended to convey from its originator to its intended recipients, one information object per attribute value.

Copy Recipients
Recipient specifiers for the IPM's copy recipients. The term ''copy recipient'' is not precisely defined by this document, but rather is given meaning by users. This attribute is present if, and only if, there are copy recipients.

Note: The copy recipients, for example, might be those users to whom the IPM is conveyed for information.

Expiry Time
The date and time at which the authorising users consider the IPM to lose its validity. This attribute is present at the option of the user that originates the IPM.

Importance
Identifies the importance that the authorising users attach to the IPM. For its defined values, see Section 4.4.4 on page 79.

Incomplete Copy
Whether one or more values of the Body attribute, or all values of another attribute, are absent from (the present instance of) the IPM.

Languages
Identifies, at the originator's option, the languages used in the composition of the Body and Subject attributes. Its value may be any of the two-character language codes identified by ISO 639.2.

Obsoleted IPMs
The IPM identifiers of the IPMs that the authorising users of the present IPM consider it to obsolete. This attribute is present at the option of the user that originates the IPM.

Originator
An O/R descriptor for the IPM's originator. This attribute is present at the option of the user that originates the IPM.

Primary Recipients
Recipient specifiers for the IPM's primary recipients. The term ''primary recipient'' is not precisely defined by this document, but rather is given meaning by users. This attribute is present if, and only if, there are primary recipients.

Note: The primary recipients, for example, might be those users who are expected to act upon the IPM.

Related IPMs
The IPM identifiers of the IPMs that the authorising users of the present IPM consider related to it. The word ''related'' is not precisely defined by this document, but rather is given meaning by users. This attribute is present at the option of the user that originates the IPM.

Note: A related IPM, for example, might be one discussed in the Body attribute of the present IPM.

Replied-to IPM
The IPM identifier of the IPM to which the present IPM is a reply. This attribute is present if, and only if, the IPM is a reply.

Reply Recipients
O/R descriptors for the users whom the authorising users request (but do not demand) are among the recipients of any replies to the IPM. This attribute is present if, and only if, the desired reply recipients are other than the originator of the IPM alone.

Note: If this attribute identifies several users, the originator may include himself among them. If he elects not to do so, he will not be considered among the desired reply recipients.

Reply Time
The date and time by which the authorising users request (but do not demand) that any replies to the IPM are originated. This attribute is present at the option of the user that originates the IPM.

Sensitivity
How sensitive the authorising users consider the IPM to be. For its defined values, see Section 4.4.8 on page 79.

Subject
A textual description of the IPM's subject matter. This attribute is present at the option of the user that originates the IPM.

This IPM
The IPM identifier assigned to the IPM.

Notes:

1. An IPM makes various assertions about its own transmittal (for example, who originates the message containing it). All of these assertions are unverified and thus should be treated with suspicion.
2. In the context of forwarding, and with respect to the Obsoleted IPMs, Related IPMs and Replied-to IPM attributes, care should be taken to distinguish between the forwarding IPM and the forwarded IPM.

4.3.9 Interpersonal Notification

An instance of class *Interpersonal Notification* (IPN) is a secondary information object conveyed between users in IM. It reports to the originator of an IPM a particular recipient's receipt or non-receipt of that IPM, which is called the subject IPM. A particular recipient shall originate at most one IPN for a particular IPM, and shall do so only in accord with the Notification Request and IPM Return Requested attributes of the subject recipient specifier, the recipient specifier that designates the recipient.

The subject recipient specifier is determined by examining the recipient specifiers that are values of the subject IPM's Primary, Copy and Blind Copy Recipients attributes. These attributes are examined in the order in which they are mentioned in the preceding sentence. Within each attribute, the recipient specifiers are examined in the order in which they appear as values. The subject recipient specifier is the first one found whose Recipient attribute has as its value an O/R descriptor whose Formal Name attribute has as its value an O/R name of the recipient on whose behalf the examination is performed.

The attributes specific to this class are listed in the following table.

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
Conversion EITs	Object (EITs)	-	0-1	-	-
IPM Intended Recipient	Object (OR Descriptor)	-	0-1	-	-
IPN Originator	Object (OR Descriptor)	-	0-1	-	-
Subject IPM	Object (IPM Identifier)	-	1	-	-

Table 4-8 Attributes Specific to Interpersonal Notification

Conversion EITs
The EITs of the subject IPM at the time of its delivery to the IPN's originator. This attribute is present if, and only if, the subject IPM was converted for delivery to that user.

IPM Intended Recipient
An O/R descriptor for the intended recipient. It shall be the value of the Recipient attribute of the subject recipient specifier. This attribute is present if, and only if, the subject IPM was delivered to an alternate (not an intended) recipient.

IPN Originator
An O/R descriptor for the IPN's originator (who is either an intended or an alternate recipient of the subject IPM). It shall be the value of the Recipient attribute of the subject recipient specifier. This attribute is present at the option of the user that originates the IPN.

Subject IPM
The IPM identifier of the subject IPM.

Note: An IPN makes various assertions about its own transmittal (for example, who originates the message containing it), as well as the transmittal of the IPM to which it responds. All of these assertions are unverified and thus should be treated with suspicion.

4.3.10 IPM Identifier

An instance of class **IPM Identifier** uniquely identifies an IPM, unambiguously distinguishing it from other IPMs originated by the same user and possibly by other users.

The attributes specific to this class are listed in the following table.

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
User	Object (OR Name)	-	0-1[1]	-	-
User-relative Identifier	String (Printable)	0-64[2]	1	-	-

[1]For 1988 alone, this value's omission is discouraged.

[2]For 1988 alone, a length of zero is discouraged.

Table 4-9 Attributes Specific to IPM Identifier

User
An O/R name of the IPM's originator. This attribute is present at the client's option.

User-relative Identifier
Uniquely identifies the IPM, unambiguously distinguishing it from all other IPMs

originated by the IPM's originator.

4.3.11 ISO 6937 Text Body Part

An instance of class **ISO 6937 Text Body Part** comprises ISO 6937 text.

The attributes specific to this class are listed in the following table.

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
Repertoire	Enum (ISO 6937 Repertoire)	-	1	graphic	-
Text	String (Octet)	-	1	-	-

Table 4-10 Attributes Specific to ISO 6937 Text Body Part

Repertoire
Identifies the character set to which the text is constrained. For its defined values, see Section 4.4.5 on page 79.

Text
The sequence of lines that constitutes the text. It shall contain no control functions, except that a carriage return followed by a line feed denotes the end of a line. Each line may contain 0-80 graphic characters for guaranteed rendition.

Note: This class is not defined by X.400, but is defined by various functional standards (see the referenced **ISO/DIS 9065**).

4.3.12 Message Body Part

An instance of class **Message Body Part** represents an IPM and, optionally, its delivery envelope. Including one IPM in another in this way is called forwarding the IPM. The enclosing IPM is called the forwarding IPM, the enclosed IPM the forwarded IPM.

The attributes specific to this class are listed in the following table.

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
Envelope	Object (Delivery Envelope)	-	0-1	-	-
IPM	Object (Interpersonal Message)	-	1	-	-

Table 4-11 Attributes Specific to Message Body Part

Envelope
The delivery envelope that accompanied the forwarded IPM when it was delivered to the originator of the forwarding IPM. However, the delivery envelope's MTS Identifier attribute shall be absent, and either the Delivery Time attribute alone, or all other attributes of the delivery envelope, may (but need not) be absent. This attribute is present at the option of the user who originates the message of which the message body part is a component.

IPM
The forwarded IPM.

Notes:

1. For 1988 alone, both of the delivery envelope omissions above are discouraged.
2. It is unverified, in any sense, that the IPM and delivery envelope are genuine.

4.3.13 Mixed-mode Body Part

An instance of class **Mixed-mode Body Part** comprises a final-form document of the kind that mixed-mode Teletex and G4 Classes 2 and 3 facsimile terminals can process.

The attributes specific to this class are listed in the following table.

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
Mixed-mode Document	String (Octet)	-	0 or more	-	-

Table 4-12 Attributes Specific to Mixed-Mode Body Part

Mixed-mode Document
The final-form document. Each value of this attribute is a protocol element that contributes to a description of the document's layout structure. Each value shall follow the rules for G4 Classes 2 and 3 facsimile, which include the BER.

4.3.14 Nationally Defined Body Part

An instance of class **Nationally Defined Body Part** comprises binary data that follows the BER. The data type of the data value the data encodes is nationally defined. The instance itself does not indicate the country involved. Rather, the client is presumed to know this a priori.

The attributes specific to this class are listed in the following table.

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
National Data	String (Encoding)	-	1	-	-

Table 4-13 Attributes Specific to Nationally Defined Body Part

National Data
The binary data. It shall follow the BER.

Note: This class is intended for use in domestic communication where the country is implicitly that of the originator and all of the recipients of the IPM that contains the nationally defined body part.

4.3.15 Non-receipt Notification

An instance of class **Non-receipt Notification** (NRN) reports its originator's failure to receive, failure to accept, or delay in receiving, an IPM.

The attributes specific to this class are listed in the following table.

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
Auto-forward Comment	String (Printable)	0-256[1]	0-1	-	-
Discard Reason	Enum (Discard Reason)	-	1	no-discard	-
Non-receipt Reason	Enum (Non-receipt Reason)	-	1	ipm-discarded	-
Returned IPM	Object (Interpersonal Message)	-	0-1	-	-

[1]For 1988 alone, a length of zero is discouraged.

Table 4-14 Attributes Specific to Non-receipt Notification

Auto-forward Comment
Information pre-supplied by the NRN's originator. This attribute is present only if the Non-receipt Reason attribute has the value ipm-auto-forwarded.

Discard Reason
Why the subject IPM was discarded. For its defined values, see Section 4.4.2 on page 78.

Non-receipt Reason
Why the NRN's originator did not receive the subject IPM (after it was delivered to him). For its defined values, see Section 4.4.6 on page 79.

Returned IPM
The subject IPM. This attribute is present if, and only if, the value of the IPM Return Requested attribute of the subject recipient specifier is true and the subject IPM was not converted for delivery to the NRN's originator.

4.3.16 Office Document Architecture Body Part

An instance of class Office Document Architecture (ODA) Body Part comprises an ODA document.

The attributes specific to this class are listed in the following table.

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
Application Profile	String(Object Id)	-	1	-	-
Architecture Class	Enum(ODA Class)	-	1	-	-
ODA Document	String(Octet)	-	0 or more	-	-

Table 4-15 Attributes Specific to ODA Body Part

Application Profile
This object identifier value shall also be used in the MTA External EITs type in addition to the id-et-oda-data object identifier.

Architecture Class
Identifies the class of the ODA document. For its defined values see ODA Class enumeration.

ODA Document
The ODA document, one string element per ODA Interchange Data Element.

4.3.17 OR Descriptor

An instance of class **OR Descriptor** identifies a user.

The attributes specific to this class are listed in the following table.

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
Formal Name	Object (OR Name)	-	0-1	-	-
Free Form Name	String (Teletex[1])	0-64[2]	0-1	-	-
Telephone Number	String (Printable[1])	0-32[2]	0-1	-	-

[1]Restricted to the graphic subset of the indicated character set.

[2]For 1988 alone, a length of zero is discouraged.

Table 4-16 Attributes Specific to OR Descriptor

Formal Name
One of the user's O/R names. This attribute is present if (but not only if) the Free Form Name attribute is absent, the O/R descriptor appears in a value of the Reply Recipients attribute of an IPM, or the O/R descriptor is the value of the Recipient attribute of a recipient specifier and the conditions stated in the description of that attribute are satisfied.

Free Form Name
An informal name for the user. This attribute is present if (but not only if) the Formal Name attribute is absent.

Telephone Number
The user's telephone number. This attribute is present at the option of the user who generates the O/R descriptor.

4.3.18 Receipt Notification

An instance of class **Receipt Notification** (RN) reports its originator's receipt, or his expected and arranged future receipt, of an IPM.

The attributes specific to this class are listed in the following table.

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
Acknowledgement Mode	Enum (Acknowledgement Mode)	-	1	manual	-
Receipt Time	String (UTC Time)	0-17	1	-	-
Supplementary Receipt Info	String (Printable)	1-256	0-1	-	-

Table 4-17 Attributes Specific to Receipt Notification

Acknowledgement Mode
Identifies how the RN was originated. For its defined values, see Section 4.4.1 on page 78.

Receipt Time
The date and time at which the RN's originator received the subject IPM.

Supplementary Receipt Info
Supplementary information about receipt of the subject IPM by the RN's originator. This attribute is present at the option of the user who originates the RN.

4.3.19 Recipient Specifier

An instance of class **Recipient Specifier** identifies and may make certain requests of a recipient of an IPM.

The attributes specific to this class are listed in the following table.

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
IPM Return Requested	Boolean	-	1	false	-
Notification Request	Enum (Notification Request)	-	1	never	-
Recipient	Object (OR Descriptor)	-	1	-	-
Reply Requested	Boolean	-	1	false	-

Table 4-18 Attributes Specific to Recipient Specifier

IPM Return Requested
Whether the recipient is asked to return the IPM in any NRN.

Notification Request
The kinds of IPN requested of the recipient in the circumstances prescribed for such IPNs (see Section 4.3.9 on page 69). For its defined values, see Section 4.4.7 on page 79.

Recipient
An O/R descriptor for the recipient. If any of the other attributes makes a request of the recipient, the Formal Name attribute of the O/R descriptor shall be present.

Reply Requested
Whether a reply is requested of the recipient. A reply is one IPM sent in response to another. A user may reply to an IPM even though no reply is requested of him, and even if he is not among the IPM's recipients but rather obtains the IPM by other means. Furthermore, a user of whom a reply is requested may refrain from replying.

4.3.20 Teletex Body Part

An instance of class **Teletex Body Part** comprises a Teletex document.

The attributes specific to this class are listed in the following table.

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
Teletex Document	String (Teletex)	-	0 or more	-	-
Teletex NBPs	Object (Teletex NBPs)	-	0-1	-	-
Telex-compatible	Boolean	-	1	false	-

Table 4-19 Attributes Specific to Teletex Body Part

Teletex Document
The Teletex document, one page per attribute value.

Teletex NBPs
The NBPs of the document. This attribute shall be present if (but not only if) the IPM contains two or more Teletex body parts. If the IPM comprises only a single Teletex body part, its NBPs may (but need not) be conveyed instead by means of the EITs attribute of the message containing the IPM.

Telex-compatible
Whether the Teletex document is Telex-compatible. If its value is true, the document shall be restricted to the ITA2 (that is, Telex) character set and no line shall be longer than 69 characters. If its value is false, the document may (but need not) violate these constraints.

4.3.21 Unidentified Body Part

An instance of class Unidentified Body Part comprises data for a body part not among those tagged body parts defined in this API.

The attributes specific to this class are listed in the following table.

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
Unidentified Data	String(Encoding)	-	1	-	-
Unidentified Tag	Integer	-	1	-	-

Table 4-20 Attributes Specific to Unidentified Body Part

Unidentified Data
The BER data that is the whole body part including its leading implicit ASN.1 CONTEXT tag.

Unidentified Tag
This is the ASN.1 context tag number that differentiates this body part. This value is extracted from the first few octets of the Unidentified Data.

4.3.22 USA Nationally Defined Body Part

An instance of class USA Nationally Defined Body Part comprises binary data that follows the BER. The data type of the data value the data encodes is defined and registered by NIST and determined by the integer value.

The attributes specific to this class are listed in the following table.

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
Body Part Number	Integer	-	1	-	-
	USA Data	String(Encoding)	-	1	-

Table 4-21 Attributes Specific to USA Nationally Defined Body Part

Body Part Number
This value is assigned by NIST to universally identify a particular body part type.

USA Data
The binary data. It shall follow the BER.

4.3.23 Videotex Body Part

An instance of class **Videotex Body Part** comprises Videotex data.

The attributes specific to this class are listed in the following table.

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
Videotex Data	String (Videotex)	-	1	-	-
Videotex Syntax	Enum (Videotex Syntax)	-	1	unspecified	1988

Table 4-22 Attributes Specific to Videotex Body Part

Videotex Data
The Videotex data, which shall conform to the Videotex syntax, if any, that the Videotex Syntax attribute denotes.

Videotex Syntax
Identifies the syntax of the Videotex data, one of those defined in T.100 and T.101. For its defined values, see Section 4.4.9 on page 80.

4.4 Syntax Definitions

This section defines the IM enumeration syntaxes, that is, the syntaxes in the Enumeration group that are specific to IM.

4.4.1 Acknowledgement Mode

An instance of enumeration syntax **Acknowledgement Mode** identifies how an RN may be originated. Its value is chosen from the following:

automatic
The originator authorises RNs in blanket fashion.

manual
The originator authorises the RN individually.

4.4.2 Discard Reason

An instance of enumeration syntax **Discard Reason** indicates why an IPM may be discarded. Its value is chosen from the following:

ipm-expired
The time identified by the IPM's Expiry Time attribute arrives, and expired IPMs are being discarded.

ipm-obsoleted
The Obsoleted IPMs attribute of another IPM delivered to the recipient identifies the IPM, and obsolete IPMs are being discarded.

no-discard
The IPM is not discarded.

user-terminated
The recipient's IM subscription is terminated.

ipm-deleted
The IPM has been deleted before receipt occurred. When a message store is involved deletion occurred before its status became **processed**.

4.4.3 IA5 Repertoire

An instance of enumeration syntax **IA5 Repertoire** identifies the character set to which the text portion of an IA5 text body part is constrained. Its value is chosen from the following:

IA5
The full IA5 character set (which is nearly identical to ASCII).

ITA2
The ITA2 (that is, Telex) character set.

4.4.4 Importance

An instance of enumeration syntax **Importance** identifies the importance that an IPM's authorising users may attach to the IPM. Its value is chosen from the following: **high**, **low** or **routine**. These values are not defined by this document, but rather are given meaning by users.

4.4.5 ISO 6937 Repertoire

An instance of enumeration syntax **ISO 6937 Repertoire** identifies the character set to which the text portion of an ISO 6937 text body part is constrained. Its value is chosen from the following:

graphic
The full graphic repertoire.

part1-and-part2

teletex
The Teletex subrepertoire.

Note: The ISO Registration Authority defines these character sets in accordance with **ISO 7350, Text Communication - Registration of Graphic Character Subrepertoires, 1984**.

4.4.6 Non-receipt Reason

An instance of enumeration syntax **Non-receipt Reason** indicates why a user may not receive an IPM after its delivery to him. Its value is chosen from the following:

ipm-auto-forwarded
The IPM is automatically forwarded.

ipm-discarded
The IPM is discarded.

4.4.7 Notification Request

An instance of enumeration syntax **Notification Request** identifies the circumstances under which an IPM may provoke an IPN. Its value is chosen from the following:

always
An RN or an NRN is issued as appropriate.

never
Neither an RN nor an NRN is issued.

non-receipt
An NRN is issued, if appropriate, but an RN is not.

4.4.8 Sensitivity

An instance of enumeration syntax **Sensitivity** indicates how sensitive an IPM's authorising users may consider the IPM to be. Its value is chosen from the following:

company-confidential
The IPM should be handled according to company-defined procedures for confidential information.

not-sensitive
The IPM is not sensitive.

personal
The IPM is addressed to its intended recipients as individuals, not as professionals.

private
The IPM should be disclosed to no one other than its intended recipients.

4.4.9 Videotex Syntax

An instance of enumeration syntax **Videotex Syntax** identifies the syntax of the data portion of a Videotex body part. Its value is chosen from the following:

data-syntax-1
Data syntax 1 as defined by T.100.

data-syntax-2
Data syntax 2 as defined by T.100.

data-syntax-3
Data syntax 3 as defined by T.100.

ids
The IDS syntax as defined by T.100.

unspecified
The syntax is unspecified.

4.5 Declaration Summary

This section lists the declarations that define the portion of the C interface that deals with the IM packages.

The declarations, as assembled here, constitute the contents of a header file to be made accessible to client programmers. The header file is **<ximp.h>**. The symbols that the declarations define are the only IM symbols the service makes visible to the client.

Note: The identifier for the variable name of type OM_STRING of a class in the Interpersonal Messaging package can usually be derived using the name of the class, preceded by "IM_C_", and replacing a blank space with an underscore. To be in line with the ANSI C language limitation, some words in the class names are excepted and are abbreviated as below:

NOTIFICATION	is abbreviated as	NOTIF
BODY_PART		BD_PRT
BILATERALLY		BILAT
EXTERNALLY		EXTERN
NATIONALLY		NATIONAL

```
/* BEGIN IM PORTION OF INTERFACE */

/* SYMBOLIC CONSTANTS */

/* Class */
#define OMP_O_IM_C_BILAT_DEF_BD_PRT       "\x2A\x86\x48\xCE\x22\x09\x00"
#define OMP_O_IM_C_BD_PRT                 "\x2A\x86\x48\xCE\x22\x09\x01"
#define OMP_O_IM_C_EXTERN_DEF_BD_PRT      "\x2A\x86\x48\xCE\x22\x09\x02"
#define OMP_O_IM_C_G3_FAX_BD_PRT          "\x2A\x86\x48\xCE\x22\x09\x03"
#define OMP_O_IM_C_G4_CLASS_1_BD_PRT      "\x2A\x86\x48\xCE\x22\x09\x04"
#define OMP_O_IM_C_IA5_TEXT_BD_PRT        "\x2A\x86\x48\xCE\x22\x09\x05"
#define OMP_O_IM_C_INTERPERSONAL_MSG      "\x2A\x86\x48\xCE\x22\x09\x06"
#define OMP_O_IM_C_INTERPERSONAL_NOTIF    "\x2A\x86\x48\xCE\x22\x09\x07"
#define OMP_O_IM_C_IPM_IDENTIFIER         "\x2A\x86\x48\xCE\x22\x09\x08"
#define OMP_O_IM_C_ISO_6937_TEXT_BD_PRT   "\x2A\x86\x48\xCE\x22\x09\x09"
#define OMP_O_IM_C_MESSAGE_BD_PRT         "\x2A\x86\x48\xCE\x22\x09\x0A"
#define OMP_O_IM_C_MIXED_MODE_BD_PRT      "\x2A\x86\x48\xCE\x22\x09\x0B"
#define OMP_O_IM_C_NATIONAL_DEF_BD_PRT    "\x2A\x86\x48\xCE\x22\x09\x0C"
#define OMP_O_IM_C_NON_RECEIPT_NOTIF      "\x2A\x86\x48\xCE\x22\x09\x0D"
#define OMP_O_IM_C_OR_DESCRIPTOR          "\x2A\x86\x48\xCE\x22\x09\x0E"
#define OMP_O_IM_C_RECEIPT_NOTIF          "\x2A\x86\x48\xCE\x22\x09\x0F"
#define OMP_O_IM_C_RECIPIENT_SPECIFIER    "\x2A\x86\x48\xCE\x22\x09\x10"
#define OMP_O_IM_C_TELETEX_BD_PRT         "\x2A\x86\x48\xCE\x22\x09\x11"
#define OMP_O_IM_C_VIDEOTEX_BD_PRT        "\x2A\x86\x48\xCE\x22\x09\x12"
#define OMP_O_IM_C_ODA_BD_PRT             "\x2A\x86\x48\xCE\x22\x09\x13"
#define OMP_O_IM_C_GENERAL_TEXT_BD_PRT    "\x2A\x86\x48\xCE\x22\x09\x14"
#define OMP_O_IM_C_UNIDENTIFIED_BD_PRT    "\x2A\x86\x48\xCE\x22\x09\x15"
#define OMP_O_IM_C_USA_NAT_DEF_BD_PRT     "\x2A\x86\x48\xCE\x22\x09\x16"

/* Enumeration */

/* Acknowledgement Mode */
#define IM_MANUAL                  ( (OM_enumeration) 0 )
#define IM_AUTOMATIC               ( (OM_enumeration) 1 )

/* Discard Reason */
```

```
#define IM_NO_DISCARD               ( (OM_enumeration) -1 )
#define IM_IPM_EXPIRED              ( (OM_enumeration) 0 )
#define IM_IPM_OBSOLETED            ( (OM_enumeration) 1 )
#define IM_USER_TERMINATED          ( (OM_enumeration) 2 )
#define IM_IPM_DELETED              ( (OM_enumeration) 3)

/* IA5 Repertoire */
#define IM_ITA2                     ( (OM_enumeration) 2 )
#define IM_IA5                      ( (OM_enumeration) 5 )

/* Importance */
#define IM_LOW                      ( (OM_enumeration) 0 )
#define IM_ROUTINE                  ( (OM_enumeration) 1 )
#define IM_HIGH                     ( (OM_enumeration) 2 )

/* ISO 6937 Repertoire */
#define IM_GRAPHIC                  ( (OM_enumeration) 1 )
#define IM_PART1_AND_PART2          ( (OM_enumeration) 2 )
#define IM_TELETEX                  ( (OM_enumeration) 3 )

/* Non-receipt Reason */
#define IM_IPM_DISCARDED            ( (OM_enumeration) 0 )
#define IM_IPM_AUTO_FORWARDED       ( (OM_enumeration) 1 )

/* Notification Request */
#define IM_NEVER                    ( (OM_enumeration) 0 )
#define IM_NON_RECEIPT              ( (OM_enumeration) 1 )
#define IM_ALWAYS                   ( (OM_enumeration) 2 )

/* ODA Class */
#define IM_ODA_FORMATTED            ((OM_enumeration) 0)
#define IM_ODA_PROCESSABLE          ((OM_enumeration) 1)
#define IM_ODA_FMTTED_PROCESSABLE   ((OM_enumeration) 2)

/* Sensitivity */
#define IM_NOT_SENSITIVE            ( (OM_enumeration) 0 )
#define IM_PERSONAL                 ( (OM_enumeration) 1 )
#define IM_PRIVATE                  ( (OM_enumeration) 2 )
#define IM_COMPANY_CONFIDENTIAL     ( (OM_enumeration) 3 )

/* Videotex Syntax */
#define IM_UNSPECIFIED              ( (OM_enumeration) -1 )
#define IM_IDS                      ( (OM_enumeration) 0 )
#define IM_DATA_SYNTAX_1            ( (OM_enumeration) 1 )
#define IM_DATA_SYNTAX_2            ( (OM_enumeration) 2 )
#define IM_DATA_SYNTAX_3            ( (OM_enumeration) 3 )

/* Type */

#define IM_ACKNOWLEDGEMENT_MODE          ( (OM_type) 100 )
#define IM_APPLICATION_PROFILE           ( (OM_type) 101 )
#define IM_ARCHITECTURE_CLASS            ( (OM_type) 102 )
#define IM_AUTHORIZING_USERS             ( (OM_type) 103 )
#define IM_AUTO_FORWARD_COMMENT          ( (OM_type) 104 )
#define IM_AUTO_FORWARDED                ( (OM_type) 105 )
#define IM_BILATERAL_DATA                ( (OM_type) 106 )
#define IM_BLIND_COPY_RECIPIENTS         ( (OM_type) 107 )
#define IM_BODY                          ( (OM_type) 108 )
#define IM_BODY_PART_NUMBER              ( (OM_type) 109 )
```

```
#define IM_CHAR_SET_REG                ( (OM_type) 110 )
#define IM_CONVERSION_EITS             ( (OM_type) 111 )
#define IM_COPY_RECIPIENTS             ( (OM_type) 112 )
#define IM_DISCARD_REASON              ( (OM_type) 113 )
#define IM_ENVELOPE                    ( (OM_type) 114 )
#define IM_EXPIRY_TIME                 ( (OM_type) 115 )
#define IM_EXTERNAL_DATA               ( (OM_type) 116 )
#define IM_EXTERNAL_PARAMETERS         ( (OM_type) 117 )
#define IM_FORMAL_NAME                 ( (OM_type) 118 )
#define IM_FREE_FORM_NAME              ( (OM_type) 119 )
#define IM_G3_FAX_NBPS                 ( (OM_type) 120 )
#define IM_G4_CLASS_1_DOCUMENT         ( (OM_type) 121 )
#define IM_IMAGES                      ( (OM_type) 122 )
#define IM_IMPORTANCE                  ( (OM_type) 123 )
#define IM_INCOMPLETE_COPY             ( (OM_type) 124 )
#define IM_IPM                         ( (OM_type) 125 )
#define IM_IPM_INTENDED_RECIPIENT      ( (OM_type) 126 )
#define IM_IPM_RETURN_REQUESTED        ( (OM_type) 127 )
#define IM_IPN_ORIGINATOR              ( (OM_type) 128 )
#define IM_LANGUAGES                   ( (OM_type) 129 )
#define IM_MIXED_MODE_DOCUMENT         ( (OM_type) 130 )
#define IM_NATIONAL_DATA               ( (OM_type) 131 )
#define IM_NON_RECEIPT_REASON          ( (OM_type) 132 )
#define IM_NOTIFICATION_REQUEST        ( (OM_type) 133 )
#define IM_OBSOLETED_IPMS              ( (OM_type) 134 )
#define IM_ODA_DOCUMENT                ( (OM_type) 135 )
#define IM_ORIGINATOR                  ( (OM_type) 136 )
#define IM_PRIMARY_RECIPIENTS          ( (OM_type) 137 )
#define IM_RECEIPT_TIME                ( (OM_type) 138 )
#define IM_RECIPIENT                   ( (OM_type) 139 )
#define IM_RELATED_IPMS                ( (OM_type) 140 )
#define IM_REPERTOIRE                  ( (OM_type) 141 )
#define IM_REPLIED_TO_IPM              ( (OM_type) 142 )
#define IM_REPLY_RECIPIENTS            ( (OM_type) 143 )
#define IM_REPLY_REQUESTED             ( (OM_type) 144 )
#define IM_REPLY_TIME                  ( (OM_type) 145 )
#define IM_RETURNED_IPM                ( (OM_type) 146 )
#define IM_SENSITIVITY                 ( (OM_type) 147 )
#define IM_SUBJECT                     ( (OM_type) 148 )
#define IM_SUBJECT_IPM                 ( (OM_type) 149 )
#define IM_SUPPLEMENTARY_RECEIPT_INFO  ( (OM_type) 150 )
#define IM_TELEPHONE_NUMBER            ( (OM_type) 151 )
#define IM_TELETEX_DOCUMENT            ( (OM_type) 152 )
#define IM_TELETEX_NBPS                ( (OM_type) 153 )
#define IM_TELEX_COMPATIBLE            ( (OM_type) 154 )
#define IM_TEXT                        ( (OM_type) 155 )
#define IM_THIS_IPM                    ( (OM_type) 156 )
#define IM_UNIDENTIFIED_DATA           ( (OM_type) 157 )
#define IM_UNIDENTIFIED_TAG            ( (OM_type) 158 )
#define IM_USA_DATA                    ( (OM_type) 159 )
#define IM_USER                        ( (OM_type) 160 )
#define IM_USER_RELATIVE_IDENTIFIER    ( (OM_type) 161 )
#define IM_VIDEOTEX_DATA               ( (OM_type) 162 )
#define IM_VIDEOTEX_SYNTAX             ( (OM_type) 163 )
#define IM_GENERAL_TEXT_DATA           ( (OM_type) 164 )
/*  Value Length */

#define IM_VL_AUTO_FORWARD_COMMENT     ( (OM_value_length) 256 )
#define IM_VL_FREE_FORM_NAME           ( (OM_value_length) 64 )
```

```
#define IM_VL_SUBJECT                   ( (OM_value_length) 128 )
#define IM_VL_SUPPL_RECEIPT_INFO        ( (OM_value_length) 256 )
#define IM_VL_TELEPHONE_NUMBER          ( (OM_value_length) 32 )
#define IM_VL_USER_RELATIVE_IDENTIFIER  ( (OM_value_length) 64 )

/* END IM PORTION OF INTERFACE */
```

Chapter 5

Message Handling Packages

5.1 Summary

This chapter defines the MH 84 Package and the MH 88 Package. The first provides the functionality of MH (1984), the second that of MH (1988). In broad terms, the latter extends the functionality of the former. The entire chapter applies to both packages, except those aspects marked by the phrase *for 1988 alone*, which apply to the MH 88 Package only. The chapter is to be understood in the context provided by the referenced **XOM** Specification.

Throughout this chapter, the words *originator* and *recipient* refer to the roles that various users, by means of their UAs, play in the conveyance of communiques and reports via the MTS. Furthermore, the term *subject message*, used with reference to a communique, denotes either that communique if it is a message, or any of the messages denoted by the communique if it is a probe.

5.2 Class Hierarchy

This section depicts the hierarchical organisation of the MH classes. Subclassification is indicated by indentation. The names of abstract classes are in italics. Thus, for example, Message is an immediate subclass of Communique, an abstract class. The names of classes to which the *om_encode*( ) function applies are in bold. The *om_create*( ) function applies to all concrete classes.

Each class with either of [A] or [G] symbol adjacent to it are useful only when using either the MA or MT interface, respectively. Other classes without the symbols are useful in both.

Object (defined in the referenced **XOM** Specification)

- Algorithm
  - — Algorithm and Result
- *Token*
  - — Asymmetric Token
- Bilateral Information
- *Content*
  - — General Content
- Delivered Message [A]
- Delivery Confirmation [A]
  - — Local Delivery Confirmation [A]
- EITs
- Expansion Record
- Extensible Object
  - — *Communique* [G]
    - — Message [G]
    - — Probe [G]
  - — *Delivered Per-recipient Report* [A]
    - — Delivered Per-recipient DR [A]
    - — Delivered Per-recipient NDR [A]
  - — *Delivery Report*
    - — Delivered Report [A]
    - — Report [G]
  - — Delivery Envelope
  - — *Per-recipient Report* [G]
    - — Per-recipient DR [G]
    - — Per-recipient NDR [G]
  - — *RD*
    - — Submitted Probe RD [A]
      - — Probe RD [G]
      - — Submitted Message RD [A]
        - — Message RD [G]
  - — Submission Results [A]
  - — *Submitted Communique* [A]
    - — Submitted Message [A]
    - — Submitted Probe [A]
- Extension
- External Trace Entry [G]
- G3 Fax NBPs
- Internal Trace Entry [G]
- Local Delivery Confirmations [A]
- Local NDR [A]

- Local Per-recipient NDR [A]
- MTS Identifier
- **OR Address**
  — **OR Name**
- Redirection Record
- Security Label
- Session
- Teletex NBPs
- *Token Public Data*
  — MT Public Data

Note: The identifier for the variable name of type OM_STRING of a class in the Message Handling package can usually be derived using the name of the class, preceded by ‘‘MH_C_’’, and replacing a blank space with an underscore. To be in line with the ANSI C language limitation, some words in the class names are excepted and are abbreviated as below:

BILATERAL_INFORMATION	is abbreviated to	BILATERAL_INFO,
DELIVERED		DELIV
CONFIRMATION		CONFIRM
CONFIRMATIONS		CONFIRMS
PER_RECIPIENT_		PER_RECIP_
DELIV_PER_RECIP_REPORT		DELIV_PER_RECIP_REP

5.3 Class Definitions

This section defines the MH classes. It describes the attributes specific to a class in a table like those used in the referenced **XOM** Specification). The table includes, however, an additional column that identifies the attributes that are for 1988 alone.

A class all the attributes specific to which are for 1988 alone is called a ''1988 class''. A 1988 class is not in the MH 84 Package, although it may be in its closure.

Note: The value length restrictions were introduced to X.400 by means of the **X.400-Series Implementors' Guide** (see Referenced Documents).

5.3.1 Algorithm

An instance of class **Algorithm** identifies a mathematical (typically cryptographic) algorithm.

The attributes specific to this class are listed in the following table.

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
Algorithm Datum	*any*	-	0-1	-	1988
Algorithm ID	String (Object Identifier)	-	1	-	1988

Table 5-1 Attributes Specific to Algorithm

Algorithm Datum
Any datum the algorithm may require as a parameter.

Algorithm ID
Identifies the algorithm generically. Its values may be drawn from an international register of algorithms or defined bilaterally.

5.3.2 Algorithm and Result

An instance of class **Algorithm and Result** identifies a mathematical (typically cryptographic) algorithm and provides its result on a particular occasion.

The attributes specific to this class are listed in the following table.

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
Algorithm Result	String (Bit)	-	1	-	1988

Table 5-2 Attributes Specific to Algorithm and Result

Algorithm Result
The result of executing the algorithm on a particular occasion.

5.3.3 Asymmetric Token

An instance of class **Asymmetric Token** conveys protected security-related information from its originator to its recipient. It provides for authentication of public information and for both authentication and confidentiality of secret information.

The attributes specific to this class are listed in the following table.

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
Public Information	Object (Token Public Data)	-	0-1	-	1988
Recipient Name	Object (OR Name)	-	1	-	1988
Secret Information	Object (Algorithm and Result)	-	0-1	-	1988
Signature	Object (Algorithm and Result)	-	1	-	1988
Time	String (UTC Time)	0-17	1	-	1988

Table 5-3 Attributes Specific to Asymmetric Token

Public Information
The public security-related information, if any, that the asymmetric token protects. The algorithm involved is applied to an instance of the MT Public Data class.

Recipient Name
The O/R name of the asymmetric token's recipient.

Secret Information
The result of encrypting the secret security-related information, if any, that the asymmetric token protects. The information is encrypted using the public asymmetric encryption key (PAEK) of the token's recipient. The algorithm involved is applied to an instance of the MT Secret Data class.

Signature
An asymmetrically encrypted, hashed version of the other class-specific attributes, computed by the originator of the asymmetric token using its secret asymmetric encryption key (SAEK). The algorithm involved is applied to an instance of the present class, from which the present attribute, however, is omitted.

Time
The date and time at which the asymmetric token was generated.

5.3.4 Bilateral Information

An instance of class **Bilateral Information** comprises binary data that follows the BER. The data type of the data value that the data encodes is defined by the ''source domain'', the management domain (MD) that supplies the data. For 1984 the source domain shall be an administration management domain (ADMD). For 1988 it may be either an ADMD or a private management domain (PRMD). The instance itself indicates neither the syntax nor the semantics of the data. Rather, the one or more MDs (ADMDs or PRMDs) for which the data is intended are presumed to know these a priori.

The attributes specific to this class are listed in the following table.

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
ADMD Name	String (Printable)	0-16	1	*see note*[1]	-
Country Name	String (Printable)	2-3	1	*see note*[1]	-
Information	any	3-1024	1	-	-
PRMD Identifier	String (Printable)	1-16	0-1	*see note*[1]	1988

[1]The value of the current session's attribute of the same name.

Table 5-4 Attributes Specific to Bilateral Information

ADMD Name

The name of the source domain, if an ADMD, or of the ADMD to which the source domain is attached, if a PRMD. It identifies the ADMD relative to the country that the Country Name attribute denotes. Its values are defined by that country.

Country Name

The name of the country of the ADMD that the ADMD Name attribute denotes. Its defined values are the numbers X.121 assigns to the country, or the character pairs ISO 3166 assigns to it.

Information

The binary data. It shall follow the BER.

PRMD Identifier

The identifier of the source domain, if a PRMD. It identifies the PRMD relative to the ADMD that the ADMD Name attribute denotes. Its values are defined by that ADMD. This attribute is present if, and only if, the source domain is a PRMD.

Note: The defined values of this attribute may (but need not) be identical to those of the PRMD Name attribute (if any) of an O/R name of a user served by the source domain.

5.3.5 Communique

An instance of class *Communique* is a primary information object conveyed by users via the MTS.

The attributes specific to this class are listed in the following table.

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
Alternate Recipient Allowed	Boolean	-	1	false	-
Bilateral Information	Object (Bilateral Information)	-	0-512[1]	-	-
Content Correlator	any	3-512	0-1	-	1988
Content Identifier	String (Printable)	1-16	0-1	-	-
Content Type	Integer or String (Object Identifier)[2]	-	1	P2-1984	-
Conversion Loss Prohibited	Boolean	-	1	false	1988
Conversion Prohibited	Boolean	-	1	false	-
Expansion History	Object (Expansion Record)	-	0-512	-	1988
Expansion Prohibited	Boolean	-	1	false	1988
External Trace Info	Object (External Trace Entry)	-	0-512[5]	-	-
Internal Trace Info	Object (Internal Trace Entry)	-	0-512	-	-
MTS Identifier	Object (MTS Identifier)	-	1	*see note*[3]	-
Origin Check	Object (Algorithm and Result)	-	0-1	-	1988
Original EITs	Object (EITs)	-	0-1	-	-
Originator Certificate	Object (Certificates)[4]	-	0-1	-	1988
Originator Name	Object (OR Name)	-	1	-	-
Reassignment Prohibited	Boolean	-	1	false	1988
Recipient Descriptors	Object (RD)	-	1-32767	-	-
Security Label	Object (Security Label)	-	0-1	-	1988

[1] For 1984 the maximum value number is 8. For 1988 it is 512.

[2] For 1984 the syntax is Integer. For 1988 it is Integer or String (Object Identifier).

[3] If no value is provided by the client, then, on taking responsibility for the object, the service will generate a value on the client's behalf.

[4] As defined in the referenced **XDS** Specification).

[5] The receiving application will receive at least one. If client does not provide an external trace entry, the service must do so.

Table 5-5 Attributes Specific to Communique

Alternate Recipient Allowed

Whether the originator permits the MTS to deliver the subject message to an alternate recipient. An MD may (but need not) assign a user, the alternate recipient, to accept delivery of messages whose Recipient Descriptors attributes contain O/R names that are invalid but recognised as meant to denote users of that MD.

Bilateral Information

Information intended for MDs through which the communique may be transferred. This attribute is present at the option of the MTA that originates the communique.

Content Correlator

Information facilitating the correlation with the communique of any reports it may provoke. This attribute is present at the option of the originator's UA. It is conveyed to recipients at delivery.

Content Identifier
Information facilitating the correlation with the communique of any reports it may provoke. This attribute is present at the option of the originator's UA. It is not conveyed to recipients at delivery.

Content Type
Identifies the syntax and semantics of the value of the Content attribute of the subject message.

Among its defined integer values are the following:

- **external**, which is reserved for interworking between 1984 and 1988 systems as prescribed by X.419.
- **P2-1984**, meaning that the Content attribute encodes an IPM or IPN in accord with the P2 rules of X.420 (1984), which include the BER.
- **P2-1988**, meaning that it encodes an IPM or IPN in accord with the P2 rules of X.420 (1988), which include the BER.
- **unidentified**, meaning that its syntax and semantics are unidentified.

The value **unidentified** and other values, not listed here, may be used if the originator and recipients so agree multilaterally.

Note: The value **unidentified** was introduced to X.400 (1984) by means of the **X.400-Series Implementors' Guide** (see Referenced Documents).

Among its defined object identifier values are the following:

- **inner-message**, denoting the object identifier specified in ASN.1 as *{joint-iso-ccitt mhs-motis(6) mts(3) 3 1}* and meaning that the Content attribute encodes an instance of the Submitted Message class in accord with the rules of X.400 (1988), which include the BER.

  The originator intends that the recipients of the outer message forward the inner message to the recipients it designates.

- **unidentified**, denoting the object identifier specified in ASN.1 as *{joint-iso-ccitt mhs-motis(6) mts(3) 3 0}* and having the same meaning as the like-named integer value.

  Note: The originator may secure the inner message (against examination, modification, or both) by securing the Content attribute of the outer message.

Conversion Loss Prohibited
Whether the originator prohibits the MTS from converting the subject message (should such conversion be necessary) if it would cause loss of information as defined in X.408.

Conversion Prohibited
Whether the originator prohibits the MTS from converting the subject message (should such conversion be necessary) under any circumstances.

Expansion History
A record of each distribution list, (DL), expansion that sought to add recipients to the communique. The records appear in chronological order.

Expansion Prohibited
Whether the originator instructs the MTS to issue an NDR rather than expand a DL if the O/R name specified for any of the recipients proves to denote a DL not a user.

External Trace Info
External trace entries which document how the communique was acted upon by each and

every MD that transferred it. The entries appear in chronological order, the first being made by the MD that originated the communique.

A single MD may add one or several trace entries to the attribute as follows. If it simply transfers the communique to the MD that is its first choice, it adds to the attribute a single trace entry whose Action attribute has the value **relayed**. Otherwise it adds one or more trace entries whose Action attributes have the value **rerouted**; these several trace entries may indicate attempts to transfer the communique to one or several MDs.

Note: This attribute provides, among other things, a basis for loop detection.

Internal Trace Info

Internal trace entries which document how the communique was acted upon by each and every MTA that transferred it. However, at the service's option (and in common practice), the scope of this attribute may be limited to the local MD, except that each entry documenting expansion or redirection is global in scope. The entries appear in chronological order.

A single MTA may add one or several trace entries to the attribute as follows. If it simply transfers the communique to the MTA that is its first choice, it adds to the attribute a single trace entry whose Action attribute has the value **relayed**. Otherwise it adds one or more trace entries whose Action attributes have the value **rerouted**; these several trace entries may indicate attempts to transfer the communique to one or several MTAs. Additionally, if it reassigns the communique to an alternate recipient, the MTA adds to the attribute a trace entry whose Action attribute has the value **redirected**.

Notes:

1. This attribute provides, among other things, a basis for loop detection.
2. This attribute is defined by MOTIS and X.400 (1988) but not by X.400 (1984).

MTS Identifier

The communique's MTS identifier. Should the communique be replicated for routing to recipients in different locations, each copy shall bear the MTS identifier of the original.

Origin Check

A means by which a third party (for example, a user or an MTA) can verify the communique's origin. This attribute is present at the option of the originator's UA. The algorithm involved is applied to an instance of the Origin Check Basis class.

Original EITs

The EITs of the Content attribute of the communique when it was submitted. This attribute is present at the option of the originator's UA.

Originator Certificate

The originator's certificate. Generated by a trusted source (for example, a certification authority (CA)), it constitutes a verified copy of the originator's PAEK. This attribute is present at the option of the originator's UA.

Originator Name

The O/R name of the communique's originator.

Reassignment Prohibited

Whether the originator prohibits the intended recipients from redirecting the communique.

Recipient Descriptors

The RDs of the communique's intended recipients.

Security Label
The security label associated with the communique. It shall be assigned in line with the security policy in force.

5.3.6 Content

An instance of class *Content* is the information that a message is intended to convey to its recipients.

The attributes specific to this class are none.

If the General Content feature has been negotiated for the session then a General Content value of this attribute is always returned. This attribute can then be used, for example, as an attribute in objects of classes from the SM 88 Package.

Note: The purpose of this abstract class is to provide a common superclass for General Content (defined in this chapter), Interpersonal Message and Interpersonal Notification (defined in the previous chapter), and message content type-dependent classes that may be defined in other specifications, for MT applications other than IM. The value of the Content attribute specific to class Message, for example, is specified to be an instance of the Content class. In a particular message, since Content is abstract, the object may be an instance of any concrete subclass of Content.

5.3.7 Delivered Message

An instance of class **Delivered Message** is a primary information object delivered by the MTS to users.

The attributes specific to this class are listed in the following table.

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
Content	Object (Content)	-	1	-	-
Envelopes	Object (Delivery Envelope)	-	1-32767	-	-

Table 5-6 Attributes Specific to Delivered Message

Content
The arbitrary binary information that the delivered message was intended to convey to its recipients.

Envelopes
The delivery envelopes of the delivered message, one for each of one or more of the users to which the message was, or is to be, delivered.

5.3.8 Delivered Per-recipient DR

An instance of class **Delivered Per-recipient DR** gives, to the originator of a submitted communique, information about the successful delivery or the deliverability of the subject message to a particular recipient.

The attributes specific to this class are listed in the following table.

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
Delivery Point	Enum (Delivery Point)	-	1	-	-
Delivery Time	String (UTC Time)	0-17	1	-	-
Proof of Delivery	Object (Algorithm and Result)	-	0-1	-	1988
Recipient Certificate	Object (Certificates)[1]	-	0-1	-	1988

[1]As defined in the referenced **XDS** Specification).

Table 5-7 Attributes Specific to Delivered Per-recipient DR

Delivery Point
The nature of the functional entity by means of which the subject message was or would have been delivered to the recipient. For its defined values, see Section 5.4.4 on page 144.

Delivery Time
The date and time at which the subject message was or would have been delivered to the recipient.

Proof of Delivery
Proof that the message has been delivered to the recipient. It is present if the originator requested proof of delivery. The algorithm involved is applied to an instance of the Proof of Delivery Basis class.

Recipient Certificate
The recipient's certificate. Generated by a trusted source (for example, a CA), it constitutes a verified copy of the recipient's PAEK. It is present if the originator requested proof of delivery and an asymmetric encryption algorithm was used to compute the proof.

5.3.9 Delivered Per-recipient NDR

An instance of class **Delivered Per-recipient NDR** gives, to the originator of a submitted communique, information about the unsuccessful delivery or the undeliverability of the subject message to a particular recipient.

The attributes specific to this class are listed in the following table.

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
Non-delivery Diagnostic	Enum (Diagnostic)	-	1	-	-
Non-delivery Reason	Enum (Reason)	-	1	-	-

Table 5-8 Attributes Specific to Delivered Per-recipient NDR

Non-delivery Diagnostic
Why in detail the subject message was not, or would not, have been conveyed to the recipient. For its defined values, see Section 5.4.5 on page 144.

Non-delivery Reason
Identifies the factor that prevented, or would have prevented, the subject message from being conveyed to the recipient. For its defined values, see Section 5.4.10 on page 148.

5.3.10 Delivered Per-recipient Report

An instance of class *Delivered Per-recipient Report* gives, to the originator of a submitted communique, information about the successful or unsuccessful delivery, or the deliverability or undeliverability, of the subject message to a particular recipient.

The attributes specific to this class are listed in the following table.

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
Actual Recipient Name	Object (OR Name)	-	1	-	-
Converted EITs	Object (EITs)	-	0-1	-	-
Forwarding Address	Object (OR Name)	-	0-1	-	1988
Originally Intended Recip	Object (OR Name)	-	0-1	-	-
Redirection History	Object (Redirection Record)	-	0-512	-	1988
Supplementary Info	String (Printable)	1-256[1]	0-1	-	-

[1]For 1984 the maximum value length is 64. For 1988 it is 256.

Table 5-9 Attributes Specific to Delivered Per-recipient Report

Actual Recipient Name
The O/R name of the recipient to which the delivered per-recipient report pertains. If the report concerns a submitted message (not a submitted probe) and the recipient is an alternate recipient, this attribute's value is the O/R name of that alternate recipient.

Converted EITs
The EITs that characterise, or would characterise, the subject message after its conversion. This attribute is present if, and only if, the MTS converted, or would have converted, the subject message.

Forwarding Address
The new postal O/R address of the recipient, a physical delivery system (PDS) patron. It is present only if the originator requested the recipient's physical forwarding address.

Originally Intended Recip
The O/R name of the originally intended recipient. This attribute is present if, and only if, the Actual Recipient Name attribute denotes an alternate recipient.

Redirection History
Documents the communique's redirection. The redirection records appear in chronological order.

Supplementary Info
Supplementary information concerning the fate of the submitted communique. This attribute is present at the option of the MD that produces the delivered per-recipient report.

Note: This attribute may be used, for example, by a Teletex access unit (AU), or Teletex/Telex conversion facility. In such cases, it may provide the received answer-back, the Telex transmission duration, or the note and received recorded message.

5.3.11 Delivered Report

An instance of class **Delivered Report** gives information to the originator of a submitted communique about the successful or unsuccessful delivery of the message to a particular recipient.

The attribute specific to this class is listed in the following table.

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
Per-recipient Reports	Object (Delivered Per-recipient Report)	-	1-32767	-	-

Table 5-10 Attributes Specific to Delivered Report

Per-recipient Reports
Delivered per-recipient reports, one for each recipient to which the delivered report applies.

5.3.12 Delivery Confirmation

An instance of class **Delivery Confirmation** is an acknowledgement of the delivery of a message or report to the UA of one of its recipients.

The attributes specific to this class are listed in the following table.

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
Proof of Delivery	Object (Algorithm and Result)	-	0-1	-	1988
Recipient Certificate	Object (Certificates)[1]	-	0-1	-	1988

[1]As defined in the referenced **XDS** Specification).

Table 5-11 Attributes Specific to Delivery Confirmation

Proof of Delivery
Proof that the message has been delivered to the recipient. It is present if the originator requested it. The algorithm involved is applied to an instance of the Proof of Delivery Basis class.

Recipient Certificate
The recipient's certificate. Generated by a trusted source (for example, a CA), it constitutes a verified copy of the recipient's PAEK. It is present if the originator requested proof of delivery and an asymmetric encryption algorithm was used to compute the proof.

5.3.13 Delivery Envelope

An instance of class **Delivery Envelope** comprises the information that the MTS conveys to a particular recipient of a message in addition to its content.

The attributes specific to this class are listed in the following table.

Table 5-12 Attributes Specific to Delivery Envelope

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
Actual Recipient Name	Object (OR Name)	-	0-1[6]	-	-
Bureau FAX Delivery	Boolean	-	0-1[5]	false	1988
Confidentiality Algorithm	Object (Algorithm)	-	0-1	-	1988
Content Identifier	String (Printable)	1-16	0-1	-	1988
Content Type	Integer or String (Object Identifier)[1]	-	0-1[6]	P2-1984	-
Conversion Loss Prohibited	Boolean	-	0-1[5]	false	1988
Conversion Prohibited	Boolean	-	0-1[6]	false	-
Converted EITs	Object (EITs)	-	0-1	-	-
Delivery Time	String (UTC Time)	0-17	0-1[5]	-	-
Distinguished Recipient Address	Object (OR Address)	-	0-1[2]	-	-
Expansion History	Object (Expansion Record)	-	0-512	-	1988
Forwarding Address Requested	Boolean	-	0-1[5]	false	1988
Forwarding Prohibited	Boolean	-	0-1[5]	false	1988
Integrity Check	Object (Algorithm and Result)	-	0-1	-	1988
MTS Identifier	Object (MTS Identifier)	-	0-1[4]	-	-
Origin Check	Object (Algorithm and Result)	-	0-1	-	1988
Original EITs	Object (EITs)	-	0-1[6]	-	-
Originally Intended Recip	Object (OR Name)	-	0-1	-	-
Originator Certificate	Object (Certificates)[3]	-	0-1	-	1988
Originator Name	Object (OR Name)	-	0-1[6]	-	-
Originator Return Address	Object (OR Address)	-	0-1	-	1988
Other Recipient Names	Object (OR Name)	-	0-32767	-	-
Postal Mode	Enum (Postal Mode)	-	0-1	-	1988
Postal Report	Enum (Postal Report)	-	0-1	-	1988
Preferred Delivery Modes	Enum (Delivery Mode)	-	0-10[5]	any	1988
Priority	Enum (Priority)	-	0-1[6]	normal	-
Proof of Delivery Requested	Boolean	-	0-1[5]	false	1988
Recipient Number for Advice	String (Teletex)	1-32	0-1	-	1988
Redirection History	Object (Redirection Record)	-	0-512	-	1988
Registration	Enum (Registration)	-	0-1	-	1988
Rendition Attributes	String (Object Identifier)	-	0-1	-	1988
Security Label	Object (Security Label)	-	0-1	-	1988
Submission Time	String (UTC Time)	0-17	0-1[6]	-	-
Token	Object (Token)	-	0-1	-	1988

[1] For 1984 the syntax is Integer. For 1988 it is Integer or String (Object Identifier).

[2] This attribute is present if, and only if, the Client Name (rather than the User Address) argument was supplied to the *MA Open*( ) function.

[3] As defined in the referenced **XDS** Specification.

[4] This attribute shall be absent when an object of this class is a subobject of a Message Body Part object, otherwise it shall be present (see Section 4.3.12 on page 71).

[5] This attribute may be absent when an object of this class is a subobject of a Message Body Part object, otherwise it shall be present (see Section 4.3.12 on page 71).

[6] When an object of this class is a subobject of a Message Body Part object (see Section 4.3.12 on page 71), this attribute may be absent ONLY when all other attributes, except Delivery Time, are absent. If any other attributes, except Delivery Time, are present, then the following attributes must be present for 1988 messages: Actual Recipient Name, Content Type, Originator Name, Priority, Submission Time.

In addition to the above, the following attributes must be present for 1984 messages, and are recommended to be present in 1988 messages for interworking with 1984 systems: Conversion Prohibited, Original EITs.

Actual Recipient Name
The recipient's O/R name.

Bureau FAX Delivery
Whether the message was to have been delivered to the recipient by Bureau FAX. This attribute shall be present only if the originator supplied a postal O/R address for the intended recipient or included physical delivery among the recipient's preferred delivery modes.

Note: Bureau FAX delivery comprises all A-H modes of delivery defined in F.170: regular delivery (A), special delivery (B), express mail (C), counter collection (D), counter collection with telephone advice (E), telefax (F), counter collection with Telex advice (G), and counter collection with Teletex advice (H).

Confidentiality Algorithm
Identifies the algorithm that the originator of the message used to encrypt its content and which the recipient may use to decrypt it.

The algorithm may be either symmetric or asymmetric. If the former, the associated key may be derived from the Token attribute or, alternatively, have been distributed by some other means. If the latter, the originator used the recipient's public key to encrypt the content, and the recipient may use the associated secret key to decrypt it.

Content Identifier
Information facilitating the correlation with the message of any reports it may have provoked. This attribute is present at the option of the originator's UA.

Content Type
Identifies the syntax and semantics of the value of the message's Content attribute. Its defined values are as prescribed for the like-named attribute specific to the Communique class.

Conversion Loss Prohibited
Whether the originator prohibited the MTS from converting the message (should such conversion have been necessary) if it would have caused loss of information as defined in X.408.

Conversion Prohibited
Whether the originator prohibited the MTS from converting the message (should such conversion have been necessary) under any circumstances.

Converted EITs
The EITs that characterise the message after its conversion. This attribute is present if, and only if, the MTS converted the message for the recipient.

Delivery Time
The date and time at which the message was delivered to the recipient.

Distinguished Recipient Address
The particular O/R address by means of which the client and service refer to the local user to whom the message is delivered.

How this address is statically chosen from among the one or more O/R addresses assigned to the user is outside the scope of this document.

Note: This attribute has no counterpart in X.400. Its purpose is to support the *start_delivery*( ) function.

Expansion History
A record of each DL expansion that sought to add recipients to the copy of the message delivered to the recipient. The records appear in chronological order.

Forwarding Address Requested
Whether the recipient's physical forwarding address is to be returned in an NDR. It shall be **true** only if the originator supplied a postal O/R address for the intended recipient or included physical delivery among the recipient's preferred delivery modes. (It may be **true** even if physical forwarding was prohibited.)

Forwarding Prohibited
Whether physical forwarding of the message was prohibited for the recipient. It shall be **true** only if the originator supplied a postal O/R address for the intended recipient or included physical delivery among the recipient's preferred delivery modes.

Integrity Check
A means by which the recipient can verify that the value of the message's Content attribute is unchanged. The algorithm involved is applied to an instance of the Integrity Check Basis class.

MTS Identifier
The message's MTS Identifier.

Origin Check
A means by which the recipient can verify the message's origin. This attribute is present at the option of the originator's UA. The algorithm involved is applied to an instance of the Origin Check Basis class.

Original EITs
The EITs of the Content attribute of the message when it was submitted. This attribute is present at the option of the originator's UA.

Originally Intended Recip
The O/R name of the originally intended recipient. This attribute is present if, and only if, the Actual Recipient Name attribute denotes an alternate recipient.

Originator Certificate
The originator's certificate. Generated by a trusted source (for example, a CA), it constitutes a verified copy of the originator's PAEK. This attribute is present at the option of the originator's UA.

Originator Name
The O/R name of the message's originator.

Originator Return Address
The postal O/R address of the message's originator. It shall be present if the originator supplied a postal O/R address for an intended recipient or included physical delivery among a recipient's preferred delivery modes. It may also be present if a recipient DL contained, or was likely to contain, one or more members for whom physical delivery was

required.

Other Recipient Names
The O/R names of the recipients, if any, of the message other than that denoted by the O/R name that is the value of the Actual Recipient Name or Intended Recipient Name attribute. This attribute is present if, and only if, the value of the message's Disclosure Allowed attribute was **true**.

Postal Mode
Identifies the kind of physical delivery to have been employed for the recipient. For its defined values, see Section 5.4.7 on page 147.

This attribute shall be present only if the originator supplied a postal O/R address for the intended recipient or included physical delivery among the recipient's preferred delivery modes. If no value is present, the value **ordinary-mail** is implied.

Postal Report
Identifies the kind of physical delivery report to be issued for the recipient. For its defined values, see Section 5.4.8 on page 148.

This attribute shall be present only if the originator supplied a postal O/R address for the intended recipient or included physical delivery among the recipient's preferred delivery modes. If no value is present, the value **undeliverable-mail-via-pds** is implied.

Preferred Delivery Modes
Identifies the modes of delivery the originator requested for the recipient. It does so in the same manner as does the like-named attribute specific to the Submitted Probe RD class.

Priority The relative priority at which the message was transferred. For its defined values, see Section 5.4.9 on page 148.

Proof of Delivery Requested
Whether the originator of the message requires proof of its delivery to the recipient.

Recipient Number for Advice
The recipient's telephone, Telex or Teletex number. It shall be present if the originator supplied a postal O/R address for the intended recipient or included physical delivery among the recipient's preferred delivery modes, and either the Postal Mode attribute specifies **cc-with-advice** or the Bureau FAX Delivery attribute is **true**.

Redirection History
Documents the communique's redirection in the course of its conveyance to the recipient. The redirection records appear in chronological order.

Registration
Identifies the kind of registered mail service to be employed in the message's physical delivery to the recipient. For its defined values, see Section 5.4.12 on page 149.

This attribute shall be present only if the originator supplied a postal O/R address for the intended recipient or included physical delivery among the recipient's preferred delivery modes. If no value is present the value **unregistered-mail** is implied.

Rendition Attributes
Identifies the message's rendition attributes. It may be present if the originator supplied a postal O/R address for the intended recipient or included physical delivery among the recipient's preferred delivery modes.

Among the attribute's defined values is **basic-rendition**, meaning a physical delivery access unit's (PDAU) basic or default rendition and denoting the object identifier specified in

ASN.1 as *{joint-iso-ccitt mhs-motis(6) mts(3) 5 0}*. MDs may use other values by bilateral agreement.

Security Label
The security label associated with the message. It shall be assigned in line with the security policy in force.

Submission Time
The date and time at which the message was submitted.

Token
The token prepared for the recipient. The algorithm involved is applied to an instance of the Asymmetric Token class.

5.3.14 Delivery Report

An instance of class **Delivery Report** is a secondary information object delivered by the MTS to users. It reports the successful or unsuccessful delivery, or the deliverability or undeliverability, of the subject message to some or all of its recipients. A single delivery report may report both successful and unsuccessful delivery, or both deliverability and undeliverability (to different recipients).

The attributes specific to this class are listed in the following table.

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
Content	Object (Content)	-	0-1	-	-
Content Correlator	*any*	3-512	0-1	-	1988
Content Extensions	Object(extension)	-	0 or more	-	1988
Content Identifier	String (Printable)	1-16	0-1	-	-
Content Type	Integer or String (Object Identifier)	-	0-1[1]	-	1988
Distinguished Recipient Address	Object (OR Address)	-	0-1[2]	-	-
Orig & Expansion History	Object (Expansion Record)	-	2-512	-	1988
Origin Check	Object (Algorithm and Result)	-	0-1	-	1988
Original EITs	Object (EITs)	-	0-1	-	1988
Reporting DL Name	Object (OR Name)	-	0-1	-	1988
Reporting MTA Certificate	Object (Certificates)[3]	-	0-1	-	1988
Security Label	Object (Security Label)	-	0-1	-	1988
Subject MTS Identifier	Object (MTS Identifier)	-	1	-	-

[1] For 1984, the attribute shall be absent. For 1988 it shall be optional.

[2] This attribute is present if, and only if, the Client Name (rather than the User Address) argument was supplied to the *MA Open*( ) function.

[3] As defined in the referenced **XDS** Specification).

Table 5-13 Attributes Specific to Delivery Report

Content
The arbitrary binary information that the submitted message was intended to convey to its recipients. The like-named attribute of that message. This attribute is present if, and only if, the submitted message's Content Return Requested attribute was **true** and among the values of the Per-recipient Reports attribute is a delivered per-recipient NDR whose Converted EITs attribute is absent.

Content Correlator
Information facilitating the correlation of the delivery report with the submitted communique that provoked it. The like-named attribute of that communique.

Content Extensions
Requests for extended processing as part of the object's submission, transfer or delivery. Only extended processing introduced to the Report Content in x.400 (1992) or later versions may be requested.

Content Identifier
Information facilitating the correlation of the delivery report with the submitted communique that provoked it. The like-named attribute of that communique.

Content Type
Identifies the syntax and semantics of the Content attribute of the submitted message. The like-named attribute of that message.

Distinguished Recipient Address
The particular O/R address by means of which the client and service refer to the local user to whom the report is delivered.

How this address is statically chosen from among the one or more O/R addresses assigned to the user is outside the scope of this document.

Note: This attribute has no counterpart in X.400. Its purpose is to support the *start_delivery*( ) function.

Orig & Expansion History
A record of each DL expansion that sought to add recipients to the copy of the submitted message delivered to the recipients to which the delivery report applies. The first record, however, documents the communique's submission, and the O/R name it contains is that of the submitted communique's originator. The other records appear in chronological order.

Origin Check
A means by which a third party (for example, a user or an MTA) can verify the delivery report's origin. The attribute may be generated by the reporting MTA when a *message (or probe) origin authentication check* was present in the submitted message. The algorithm involved is applied to an instance of the Origin Check Basis class.

Original EITs
The EITs of the Content attribute of the submitted communique when it was submitted. This attribute is present if the content of the subject message of the submitted communique was converted, or would have been converted, for delivery to one or more of the recipients to which the delivery report applies.

Reporting DL Name
The O/R name of the DL, if any, whose expansion point forwarded the delivery report to the DL's owner, in line with the DL's reporting policy.

Reporting MTA Certificate
The certificate of the MTA that originated the delivery report. Generated by a trusted source (for example, a CA), it constitutes a verified copy of the MTA's PAEK. It is present if the Origin Check attribute is present.

Security Label
The security label associated with the delivery report. The like-named attribute of the submitted communique.

Subject MTS Identifier
The MTS identifier of the submitted communique.

5.3.15 EITs

An instance of class **EITs** summarises the encoded information types (EITs) represented in a message's Content attribute. It provides a basis for deciding whether the message can be delivered without conversion.

The attributes specific to this class are listed in the following table.

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
Builtin EITs	Enum (Builtin EIT)	-	0-10	IA5-text	-
External EITs	String (Object Identifier)	-	0-1024	-	1988
G3 Fax NBPs	Object (G3 Fax NBPs)	-	0-1	-	-
G4 Fax NBPs	String (Encoding)	see note[1]	0-1	-	-
Teletex NBPs	Object (Teletex NBPs)	-	0-1	-	-

[1]As prescribed for G4 facsimile.

Table 5-14 Attributes Specific to EITs

Builtin EITs
With the External EITs attribute, identifies the kinds of information in the Content attribute. This attribute is present at the client's option. For its defined values, see Section 5.4.2 on page 143.

External EITs
With the Builtin EITs attribute, identifies the kinds of information in the Content attribute. This attribute is present at the client's option.

Among the attribute's defined values are those, in the following table, which have the same meanings as the like-named values of the Builtin EITs attribute. The first column of the table lists the symbolic values. The second column specifies in ASN.1 the object identifiers that the values denote.

Value	Object Identifier (ASN.1)
g3-fax	{joint-iso-ccitt(2) mhs-motis(6) mts(3) 4 3}
g4-class-1	{joint-iso-ccitt(2) mhs-motis(6) mts(3) 4 4}
ia5-text	{joint-iso-ccitt(2) mhs-motis(6) mts(3) 4 2}
mixed-mode	{joint-iso-ccitt(2) mhs-motis(6) mts(3) 4 9}
teletex	{joint-iso-ccitt(2) mhs-motis(6) mts(3) 4 5}
telex	{joint-iso-ccitt(2) mhs-motis(6) mts(3) 4 1}
undefined	{joint-iso-ccitt(2) mhs-motis(6) mts(3) 4 0}
videotex	{joint-iso-ccitt(2) mhs-motis(6) mts(3) 4 6}

Table 5-15 Selected Values of the External EITs Attribute

Note: The interfaces do not support X.400's Simple Formattable Document (SFD) and Voice EITs. X.400 (1984) does not fully specify the Voice type. X.400 (1988) abandons the SFD type, and leaves the Voice type only partially specified.

G3 Fax NBPs

A G3 fax NBPs that further characterises the G3 facsimile images. This attribute is present only if **g3-fax** is among the values of either the Builtin EITs or the External EITs attribute.

Note: For 1988 the use of this attribute is denigrated. It is suggested that a value of the External NBPs attribute is defined for each set of NBPs of interest.

G4 Fax NBPs

The NBPs that further characterise the final-form documents. These NBPs correspond to the presentation capabilities (control and graphic character sets, resolution, etc.) of T.400, T.503 and T.501. The attribute's value shall follow the rules for signalling such capabilities, which include the BER. This attribute is present only if **g4-class-1** or **mixed-mode** is among the values of either the Builtin EITs or the External EITs attribute.

Note: For 1988 the use of this attribute is denigrated. It is suggested that a value of the External NBPs attribute is defined for each set of NBPs of interest.

Teletex NBPs

Teletex NBPs that further characterise the teletex documents. This attribute is present only if **teletex** is among the values of either the Builtin EITs or the External EITs attribute.

Note: For 1988 the use of this attribute is denigrated. It is suggested that a value of the External NBPs attribute is defined for each set of NBPs of interest.

If the Content attribute comprises several instances of G3 facsimile, G4 facsimile or Teletex information (constituting, for example, several parts of the body of an IPM), the G3 Fax NBPs, G4 Fax NBPs, or Teletex NBPs attribute shall indicate the logical union of the NBPs of the instances.

5.3.16 Expansion Record

An instance of class **Expansion Record** documents the expansion of a DL or, occasionally, the submission of a communique.

The attributes specific to this class are listed in the following table.

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
Name	Object (OR Name)	-	1	-	1988
Time	String (UTC Time)	0-17	1	-	1988

Table 5-16 Attributes Specific to Expansion Record

Name

If the expansion record documents the expansion of a DL, this attribute is the O/R name of the DL. Otherwise, it is that of the communique's originator.

Time

If the expansion record documents the expansion of a DL, this attribute is the date and time the DL was expanded. Otherwise, it is the date and time the communique was submitted.

5.3.17 Extensible Object

An instance of class **Extensible Object** is an object whose composition was extended in X.400 (1988) and may be extended again in subsequent versions of X.400.

The attributes specific to this class are listed in the following table.

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
Critical for Delivery	Integer	-	0 or more	*see note*[1]	1988
Critical for Submission	Integer	-	0 or more	*see note*[1]	1988
Critical for Transfer	Integer	-	0 or more	*see note*[1]	1988
Extensions	Object (Extension)	-	0 or more	-	1988

[1]The integers that denote the types whose values request extended processing that X.400 specifies is, by default, critical in the specified context.

Table 5-17 Attributes Specific to Extensible Object

Critical for Delivery
The integers that denote the types, specific to subclasses of this class, of the attributes whose values request extended processing deemed critical to the object's delivery. Only extended processing introduced in X.400 (1988) or later versions of X.400 may be identified.

Critical for Submission
The integers that denote the types, specific to subclasses of this class, of the attributes whose values request extended processing deemed critical to the object's submission. Only extended processing introduced in X.400 (1988) or later versions of X.400 may be identified.

Critical for Transfer
The integers that denote the types, specific to subclasses of this class, of the attributes whose values request extended processing deemed critical to the object's transfer. Only extended processing introduced in X.400 (1988) or later versions of X.400 may be identified.

Extensions
Requests for extended processing as part of the object's submission, transfer or delivery. Only extended processing introduced in X.400 (1992) or later versions of X.400 may be requested.

X.400 specifies the default criticality (for submission, transfer and delivery) of each form of extended processing it defines. The originator of a particular communique may deem the criticality of a particular form to be different, however.

When extended processing that is deemed critical for submission, delivery or transfer cannot be afforded a communique or report, the following actions are taken:

Submission
If it cannot provide a communique with processing that is critical for submission, the *Submit*( ) function reports an exception.

Transfer
If neither the client nor the service can provide a communique or report with processing that is critical for transfer, the service issues an NDR or discards the report, respectively.

Delivery
If neither the client nor the service can provide a communique or report with processing that is critical for delivery, the service issues an NDR or discards the report, respectively.

By means outside the scope of this document, the client informs the service of the kinds of extended transfer or delivery processing of which the client is capable. This information enables the service to act as prescribed above.

5.3.18 Extension

An instance of class **Extension** is a request for extended processing as part of the submission, transfer or delivery or a communique or report. Only extended processing introduced in X.400 (1992) or later versions of X.400 may be requested.

The attributes specific to this class are listed in the following table.

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
Critical for Delivery	Boolean	-	1	false[1]	1988
Critical for Submission	Boolean	-	1	false[1]	1988
Critical for Transfer	Boolean	-	1	false[1]	1988
Extension Type	Integer or String (Object Identifier)[2]	-	1	-	1988
Extension Value	*any*	-	0-1	-	1988

[1]The default criticality prescribed by X.400.

[2]An integer is used to denote standard forms of extended processing, an object identifier to denote proprietary forms. Only an integer is permitted upon entry to an ADMD, and only an integer is present upon exit from an ADMD.

Table 5-18 Attributes Specific to Extension

Critical for Delivery
: Whether the extended processing is deemed critical to the communique's or report's delivery.

Critical for Submission
: Whether the extended processing is deemed critical to the communique's or report's submission.

Critical for Transfer
: Whether the extended processing is deemed critical to the communique's or report's transfer.

Extension Type
: Identifies the extended processing.

Extension Value
: Parameterises the extended processing.

5.3.19 External Trace Entry

An instance of class **External Trace Entry** describes one or more actions that an MD took with respect to a communique or report.

The attributes specific to this class are listed in the following table.

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
ADMD Name	String (Printable)	0-16	1	*see note*[1]	-
Country Name	String (Printable)	2-3	1	*see note*[1]	-
PRMD Identifier	String (Printable)	1-16	0-1	*see note*[1]	-
Attempted ADMD Name	String (Printable)	0-16	0-1	-	-
Attempted Country Name	String (Printable)	2-3	0-1	-	-
Attempted PRMD Identifier	String (Printable)	1-16	0-1	-	-
Action	Enum (Action)	-	1-3	relayed	-
Arrival Time	String (UTC Time)	0-17	1	-	-
Converted EITs	Object (EITs)	-	0-1	-	-
Deferred Time	String (UTC Time)	0-17	0-1	-	-

[1]The value of the current session's attribute of the same name. For .I "PRMD Name" , the correponding attribute in the session object is PRMD Identifier.

Table 5-19 Attributes Specific to External Trace Entry

The following three attributes identify the ''tracing domain'', the MD that produced the entry.

ADMD Name
The name of the tracing domain, if an ADMD, or of the ADMD to which the tracing domain is attached, if a PRMD. It identifies the ADMD relative to the country that the Country Name attribute denotes. Its values are defined by that country.

Country Name
The name of the country of the ADMD that the ADMD Name attribute denotes. Its defined values are the numbers X.121 assigns to the country, or the character pairs ISO 3166 assigns to it.

PRMD Identifier
The identifier of the tracing domain, if a PRMD. It identifies the PRMD relative to the ADMD that the ADMD Name attribute denotes. Its values are defined by that ADMD. This attribute is present if, and only if, the tracing domain is a PRMD.

Note: The defined values of this attribute may (but need not) be identical to those of the PRMD Name attribute (if any) of an O/R name of a user served by the tracing domain.

The following three attributes identify the ''attempted domain'', the MD to which the tracing domain attempted but failed to transfer the communique or report. (These attributes have values only if the Action attribute's value is rerouted.)

Attempted ADMD Name
The name of the attempted domain, if an ADMD, or of the ADMD to which the attempted domain is attached, if a PRMD. The attribute value identifies the ADMD relative to the country that the Attempted Country Name attribute denotes. Its values are defined by that country. This attribute is present if, and only if, the Action attribute has the value **rerouted**.

Attempted Country Name
The name of the country of the ADMD that the Attempted ADMD Name attribute denotes. Its defined values are the numbers X.121 assigns to the country, or the character pairs ISO 3166 assigns to it. This attribute is present if, and only if, the Attempted ADMD Name attribute is present.

Attempted PRMD Identifier
The identifier of the attempted domain if a PRMD. It identifies the PRMD relative to the ADMD that the Attempted ADMD Name attribute denotes. Its values are defined by that ADMD. This attribute is present if, and only if, the attempted domain is a PRMD.

Note: The defined values of this attribute may (but need not) be identical to those of the PRMD Name attribute (if any) of an O/R name of a user served by the attempted domain.

The following attributes indicate how the tracing domain handled the communique or report.

Action
Identifies the routing action the tracing domain took. For its defined values, see Section 5.4.1 on page 142. The values relayed and rerouted are mutually exclusive. For 1984 only one value (either relayed or rerouted) should be present.

Arrival Time
The date and time at which the communique or report entered the tracing domain. If the tracing domain is that at which the communique or report originated, this shall be the submission time, if a communique, or the delivery or non-delivery time, if a report.

Converted EITs
The EITs that characterise, or would characterise, the subject message after its conversion. This attribute is present if, and only if, the external trace entry is for a communique (not a report) and the tracing domain converted, or would have converted, the subject message.

Deferred Time
The date and time at which the tracing domain released the message. This attribute is present if, and only if, the external trace entry is for a message (not a probe or report) and the tracing domain held the message because its originator requested deferred delivery.

5.3.20 G3 Fax NBPs

An instance of class **G3 Fax NBPs** comprises the non-basic parameters (NBPs) of a set of G3 facsimile images. Each NBP identifies a non-basic capability of a G3 facsimile terminal.

Note: In the G3 facsimile service, NBPs are conveyed between terminals by means of the three- or four-octet Facsimile Information Field (FIF) of the Digital Command Signal (DCS) data structure of T.30.

The attributes specific to this class are listed in the following table.

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
A3 Width	Boolean	-	1	false	-
B4 Length	Boolean	-	1	false	-
B4 Width	Boolean	-	1	false	-
Fine Resolution	Boolean	-	1	false	-
Two Dimensional	Boolean	-	1	false	-
Uncompressed	Boolean	-	1	false	-
Unlimited Length	Boolean	-	1	false	-

Table 5-20 Attributes Specific to G3 Fax NBPs

A3 Width
Whether the like-named G3 facsimile capability is required for the images.

B4 Length
See A3 Width.

B4 Width
See A3 Width.

Fine Resolution
See A3 Width.

Two Dimensional
See A3 Width.

Uncompressed
See A3 Width.

Unlimited Length
See A3 Width.

5.3.21 General Content

An instance of class **General Content** is the information that a message is intended to convey to its recipients, made available as binary data.

The attributes specific to this class are listed in the following table.

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
Binary Content	String (octet)	-	1	-	-

Table 5-21 Attributes Specific to General Content

Binary Content
The binary data that the message is intended to convey to its recipients. The MTS modifies this attribute's value only for purposes of conversion.

5.3.22 Internal Trace Entry

An instance of class **Internal Trace Entry** describes one or more actions that an MTA took with respect to a communique or report.

The attributes specific to this class are listed in the following table.

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
MTA Name	String (IA5)	0-32	1	-	-
ADMD name	String (Printable)	0-16	1	*see note*[1]	1988
Country Name	String (Printable)	2-3	1	*see note*[1]	1988
PRMD identifier	String (Printable)	1-16	0-1	*see note*[1]	1988
Attempted MTA Name	String (IA5)	0-32	0-1	-	-
Attempted ADMD name	String (Printable)	0-16	0-1	-	1988
Attempted Country Name	String (Printable)	2-3	0-1	-	1988
Attempted PRMD identifier	String (Printable)	1-16	0-1	-	1988
Action	Enum (Action)	-	1-3	relayed	-
Arrival Time	String (UTC Time)	0-17	1	-	-
Converted EITS	Object (EITs)	-	0-1	-	1988
Deferred Time	String (UTC Time)	0-17	0-1	-	-

[1]The value of the current session's attribute of the same name.

Table 5-22 Attributes Specific to Internal Trace Entry

The following attributes identify the ''tracing MTA'', the MTA that produced the entry.

MTA Name
The name of the tracing MTA. It identifies that MTA relative to the MD of which it is a part. Its values are defined by that MD.

ADMD Name
The name of the tracing domain, if an ADMD, or of the ADMD to which the tracing domain is attached, if a PRMD. It identifies the ADMD relative to the country that the Country Name attribute denotes. Its values are defined by that country.

Country Name
The name of the country of the ADMD that the ADMD Name attribute denotes. Its defined values are the numbers X.121 assigns to the country, or the character pairs ISO 3166 assigns to it.

PRMD Identifier
The identifier of the tracing domain, if a PRMD. It identifies the PRMD relative to the ADMD that the ADMD Name attribute denotes. Its values are defined by that ADMD. This attribute is present if, and only if, the tracing domain is a PRMD.

Note: The defined values of this attribute may (but need not) be identical to those of the PRMD Name attribute (if any) of an O/R name of a user served by the tracing domain.

The following attributes identify the ''attempted MTA'', the MTA to which the tracing MTA attempted but failed to transfer the communique or report.

Attempted MTA Name
The name of the attempted MTA. It identifies the MTA relative to the MD of which the MTA is a part. Its values are defined by that MD. This attribute is present if, and only if, the Action attribute has the value **rerouted**.

Attempted ADMD Name
The name of the attempted domain, if an ADMD, or of the ADMD to which the attempted domain is attached, if a PRMD. The attribute value identifies the ADMD relative to the country that the Attempted Country Name attribute denotes. Its values are defined by that country. This attribute is present if, and only if, the Action attribute has the value **rerouted**. This attribute is not present if *Attempted MTA name* is present.

Attempted Country Name
The name of the country of the ADMD that the Attempted ADMD Name attribute denotes. Its defined values are the numbers X.121 assigns to the country, or the character pairs ISO 3166 assigns to it. This attribute is present if, and only if, the Attempted ADMD Name attribute is present. This attribute is not present if *Attempted MTA name* is present.

Attempted PRMD Identifier
The identifier of the attempted domain if a PRMD. It identifies the PRMD relative to the ADMD that the Attempted ADMD Name attribute denotes. Its values are defined by that ADMD. This attribute is present if, and only if, the attempted domain is a PRMD. This attribute is not present if *Attempted MTA name* is present.

Note: The defined values of this attribute may (but need not) be identical to those of the PRMD Name attribute (if any) of an O/R name of a user served by the attempted domain.

The following attributes indicate how the tracing MTA handled the communique or report.

Action
Identifies the routing action that the tracing MTA took. For its defined values, see Section 5.4.1 on page 142. The values relayed and rerouted are mutually exclusive. For 1984 only one value (either relayed or rerouted) should be present.

Arrival Time
The date and time at which the communique or report entered the tracing MTA.

Converted EITs
The EITs that characterise, or would characterise, the subject message after its conversion. This attribute is present if, and only if, the external trace entry is for a communique (not a report) and the tracing domain converted, or would have converted, the subject message.

Deferred Time
The date and time at which the tracing MTA released the message. This attribute is present if, and only if, the internal trace entry is for a message (not a probe or report) and the tracing MTA held the message because its originator requested deferred delivery.

Note: Internal trace entries are part of MOTIS and X.400 (1988) but not X.400 (1984). With respect to the latter they are included in the interfaces because they are recognised by various functional standards, or profiles, based upon X.400 (1984). The scope of such recognition is outside the scope of this document.

5.3.23 Local Delivery Confirmation

An instance of class **Local Delivery Confirmation** is an acknowledgement of the delivery of a message (but not a report) to the UA of a local user.

The attributes specific to this class are listed in the following table.

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
Recipient Number	Integer	-	1	0	1988

Table 5-23 Attributes Specific to Local Delivery Confirmation

Recipient Number
The value position, within the Envelopes attribute specific to the Delivered Message class, of the delivery envelope intended for the local recipient to which the local delivery confirmation applies.

5.3.24 Local Delivery Confirmations

An instance of class **Local Delivery Confirmations** comprises confirmations of the delivery of a message (but not a report) to zero or more local recipients.

The attributes specific to this class are listed in the following table.

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
Delivery Confirmations	Object (Local Delivery Confirm.)	-	0-32767	-	1988

Table 5-24 Attributes Specific to Local Delivery Confirmations

Delivery Confirmations
A local delivery confirmation for each local recipient to which the local DR applies.

5.3.25 Local NDR

An instance of class **Local NDR** comprises information about the unsuccessful delivery of a message or a report to zero or more local recipients.

The attributes specific to this class are listed in the following table.

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
Per-recipient Reports	Object (Local Per-recipient NDR)	-	0-32767	-	-

Table 5-25 Attributes Specific to Local NDR

Per-recipient Reports
A local per-recipient NDR for each local recipient to which the local NDR applies.

5.3.26 Local Per-recipient NDR

An instance of class **Local Per-recipient NDR** comprises information about the unsuccessful delivery of a message or report to a particular local recipient.

The attributes specific to this class are listed in the following table.

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
Non-delivery Diagnostic	Enum (Diagnostic)	-	1	no-diagnostic	-
Non-delivery Reason	Enum (Reason)	-	1	transfer-failed	-
Recipient Number	Integer	-	1[1]	0	-
Supplementary Info	String(Printable)	1-256[2]	0-1	-	-
Temporary	Boolean	-	1	false	-

[1] This attribute shall only be present if the object is referring to the unsuccessful delivery of a message. It is not required for reports.

[2] For 1984 the maximum value length is 64. For 1988 it is 256.

Table 5-26 Attributes Specific to Local Per-recipient NDR

Non-delivery Diagnostic
Why in detail the subject message was not, or would not, have been conveyed to the recipient. For its defined values, see Section 5.4.5 on page 144.

This attribute cannot be passed to an MTA by means of the P3 protocol.

Non-delivery Reason
Identifies the factor that prevented, or would have prevented, the subject message from being conveyed to the recipient. For its defined values, see Section 5.4.10 on page 148.

This attribute cannot be passed to an MTA by means of the P3 protocol.

Recipient Number
The value position, within the Envelopes attribute specific to the Delivered Message class, of the delivery envelope intended for the local recipient to which the local per-recipient NDR applies.

Supplementary Information
Supplementary information concerning the fate of the message.

Temporary
Whether the factor that prevented, or would have prevented, the subject message from being conveyed to the recipient is temporary, rather than permanent.

5.3.27 Message

An instance of class **Message** is a primary information object conveyed between users by the MTS. It conveys arbitrary binary data from one user, the originator, to one or more users, the recipients.

The attributes specific to this class are listed in the following table.

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
Confidentiality Algorithm	Object (Algorithm)	-	0-1	-	1988
Content	Object (Content)	-	1	-	-
Content Return Requested	Boolean	-	1	false	-
Deferred Delivery Time	String (UTC Time)	0-17	0-1	-	-
Disclosure Allowed	Boolean	-	1	false	-
Latest Delivery Time	String (UTC Time)	0-17	0-1	-	1988
Originator Return Address	Object (OR Address)	-	0-1	-	1988
Priority	Enum (Priority)	-	1	normal	-

Table 5-27 Attributes Specific to Message

Confidentiality Algorithm

Identifies the algorithm that the originator of the message used to encrypt its content and which the recipients may use to decrypt it.

The algorithm may be either symmetric or asymmetric. If the former, the associated key may be derived from the Token attribute of any of the message's RDs or, alternatively, distributed by some other means. If the latter, the originator may use the intended recipient's public key to encrypt the content, and the recipient may use the associated secret key to decrypt it. The message must be addressed to either a single recipient or a group of recipients sharing the same key pair.

Content

The arbitrary binary information the message is intended to convey to its recipients. The MTS modifies this attribute's value only for purposes of conversion.

Content Return Requested

Whether the Content attribute is to be included as the like-named attribute of any NDRs the message provokes.

Deferred Delivery Time

The date and time, if any, before which the message shall not be delivered. Delivery deferral is normally the responsibility of the MD that originates the message. Thus messages whose Deferred Delivery Time attributes are present shall be transferred between MDs only by bilateral agreement between those MDs.

Disclosure Allowed

Whether the O/R names of other recipients are to be indicated to each recipient at delivery.

Latest Delivery Time

The date and time after which the MTS is to treat the message as undeliverable if it has not yet been delivered to a particular recipient.

Originator Return Address

The postal O/R address of the message's originator. It shall be present if the originator supplied a postal O/R address for an intended recipient or included physical delivery among a recipient's preferred delivery modes. The attribute may also be present if a recipient DL contains, or is likely to contain, one or more members for whom physical delivery is required.

Priority

The relative priority at which the message is to be transferred. For its defined values, see Section 5.4.9 on page 148.

5.3.28 Message RD

An instance of class **Message RD** identifies an intended recipient of a message and records certain information about that recipient.

The attributes specific to this class are listed in the following table.

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
MTA Report Request	Enum (Report Request[1])	-	1	non-delivery	-
MTA Responsibility	Boolean	-	1	true	-
Recipient Number	Integer	-	1	-	-
Redirection History	Object (Redirection Record)	-	0-512	-	1988

[1]The value shall not be **never**.

Table 5-28 Attributes Specific to Message RD

MTA Report Request
The kinds of report that the originating MD requests in the circumstances prescribed for such reports. For its defined values, see Section 5.4.13 on page 149.

This attribute shall neither exclude reports nor call for less external trace information than does the Originator Report Request attribute.

MTA Responsibility
Whether the message of which the RD is a part is to be delivered to the recipient.

Note: If this attribute is **false**, another copy of the message has been created and is being independently routed to this particular recipient. The present copy shall not be routed for eventual delivery to the recipient.

Recipient Number
The recipient's ordinal position in the list of recipients originally specified by the originator of the message of which the recipient specifier is a part. For the first recipient this attribute's value is one.

Redirection History
Documents the message's redirection in the course of its conveyance to the recipient. The redirection records appear in chronological order.

5.3.29 MT Public Data

An instance of class **MT Public Data** provides the means by which the message token conveys public security-related information. If the Integrity Check attribute is present, the token provides for non-repudiation of the content of the message that bears the token. If the Security Label attribute is present also, the token provides proof of the association between the content and the message's security label.

The attributes specific to this class are listed in the following table.

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
Confidentiality Algorithm	Object(Algorithm)	-	0-1	-	1988
Integrity Check	Object (Algorithm and Result)	-	0-1	-	1988
Message Sequence Number	Integer	-	0-1	-	1988
Proof of Delivery Requested	Boolean	-	1	false	1988
Security Label	Object (Security Label)	-	0-1	-	1988

Table 5-29 Attributes Specific to MT Public Data

Confidentiality Algorithm
Identifies the algorithm used by the originator of the message to encrypt the message content.

Integrity Check
A means by which any recipient of the message that bears the token can verify that its content is unchanged. The algorithm involved is applied to an instance of the Integrity Check Basis class. This attribute is present at the option of the token's originator.

Message Sequence Number
The like-named attribute, if any, of the message that bears the token.

Proof of Delivery Requested
The like-named attribute of the RD that bears the token.

Security Label
The like-named attribute, if any, of the message that bears the token.

5.3.30 MTS Identifier

An instance of class **MTS Identifier** distinguishes a communique or report from all other communiques and reports, regardless of their class, ever conveyed by the MTS. The MTS identifier is assigned by the ''originating domain'', the MD at which the communique or report is originated.

The attributes specific to this class are listed in the following table.

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
ADMD Name	String (Printable)	0-16	1	*see note*[1]	-
Country Name	String (Printable)	2-3	1	*see note*[1]	-
Local Identifier	String (IA5)	1-32	1	-	-
PRMD Identifier	String (Printable)	1-16	0-1	*see note*[1]	-

[1]The current session's attribute of the same name.

Table 5-30 Attributes Specific to MTS Identifier

ADMD Name
The name of the originating domain, if an ADMD, or of the ADMD to which the originating domain is attached, if a PRMD. It identifies the ADMD relative to the country that the Country Name attribute denotes. Its values are defined by that country.

Country Name
The name of the country of the ADMD that the ADMD Name attribute denotes. Its defined values are the numbers X.121 assigns to the country, or the character pairs ISO 3166 assigns to it.

Local Identifier
Identifies the communique or report, distinguishing it from all other communiques and reports, regardless of their class, ever originated by the originating MD.

PRMD Identifier
The identifier of the originating MD, if a PRMD. It identifies the PRMD relative to the ADMD that the ADMD Name attribute denotes. Its values are defined by that ADMD. This attribute is present if, and only if, the originating domain is a PRMD.

Note: The defined values of this attribute may (but need not) be identical to those of the PRMD Name attribute of an O/R name of a user served by the originating domain.

5.3.31 OR Address

An instance of class **OR Address** distinguishes one user or DL from another and identifies its point of access to the MTS. Every user or DL is assigned one or more MTS access points and thus one or more originator/recipient (O/R) addresses.

The attributes specific to this class are listed in the following table.

Table 5-31 Attributes Specific to OR Address

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
ADMD Name	String (Printable)	0-16	0-1	*see note*[1]	-
Common Name	String (Printable or Teletex[3])	1-64	0-2	-	1988
Country Name	String (Printable)	2-3	0-1	*see note*[1]	-
Domain Type 1	String (Printable or Teletex[3,4])	1-8	0-2[5]	-	-
Domain Type 2	String (Printable or Teletex[3,4])	1-8	0-2[5]	-	-
Domain Type 3	String (Printable or Teletex[3,4])	1-8	0-2[5]	-	-
Domain Type 4	String (Printable or Teletex[3,4])	1-8	0-2[5]	-	-
Domain Value 1	String (Printable or Teletex[3,4])	1-128	0-2[5]	-	-
Domain Value 2	String (Printable or Teletex[3,4])	1-128	0-2[5]	-	-
Domain Value 3	String (Printable or Teletex[3,4])	1-128	0-2[5]	-	-
Domain Value 4	String (Printable or Teletex[3,4])	1-128	0-2[5]	-	-
Generation	String (Printable or Teletex[3,4])	1-3	0-2[5]	-	-
Given Name	String (Printable or Teletex[3,4])	1-16	0-2[5]	-	-
Initials	String (Printable or Teletex[3,4])	1-5	0-2[5]	-	-
ISDN Number	String (Numeric)	1-15	0-1	-	1988
ISDN Subaddress	String (Numeric)	1-40	0-1[6]	-	1988
Numeric User Identifier	String (Numeric)	1-32	0-1	-	-
Organisation Name	String (Printable or Teletex[3,4])	1-64	0-2[5, 7]	-	-
Organisational Unit Name 1	String (Printable or Teletex[3,4])	1-32	0-2[5]	-	-
Organisational Unit Name 2	String (Printable or Teletex[3,4])	1-32	0-2[5]	-	-
Organisational Unit Name 3	String (Printable or Teletex[3,4])	1-32	0-2[5]	-	-
Organisational Unit Name 4	String (Printable or Teletex[3,4])	1-32	0-2[5]	-	-
Postal Address Details	String (Printable or Teletex[3])	1-30	0-2	-	1988
Postal Address in Full	String (Teletex)	1-185	0-1	-	1988

Postal Address in Lines	String (Printable)	1-30	0-6	-	1988
Postal Code	String (Printable)	1-16	0-1	-	1988
Postal Country Name	String (Printable)	2-3	0-1	-	1988
Postal Delivery Point Name	String (Printable or Teletex[3])	1-30	0-2	-	1988
Postal Delivery System Name	String (Printable)	1-16	0-1	-	1988
Postal General Delivery Address	String (Printable or Teletex[3])	1-30	0-2	-	1988
Postal Locale	String (Printable or Teletex[3])	1-30	0-2	-	1988
Postal Office Box Number	String (Printable or Teletex[3])	1-30	0-2	-	1988
Postal Office Name	String (Printable or Teletex[3])	1-30	0-2	-	1988
Postal Office Number	String (Printable or Teletex[3])	1-30	0-2	-	1988
Postal Organisation Name	String (Printable or Teletex[3])	1-30	0-2	-	1988
Postal Patron Details	String (Printable or Teletex[3])	1-30	0-2	-	1988
Postal Patron Name	String (Printable or Teletex[3])	1-30	0-2	-	1988
Postal Street Address	String (Printable or Teletex[3])	1-30	0-2	-	1988
Presentation Address	Object (Presentation Address)[2]	-	0-1	-	1988
PRMD Name	String (Printable)	1-16	0-1	*see note*[1]	-
Surname	String (Printable or Teletex[3,4])	1-40	0-2[5]	-	-
Terminal Identifier	String (Printable)	1-24	0-1	-	-
Terminal Type	Enum (Terminal Type)	-	0-1	-	1988
X121 Address	String (Numeric)	1-15	0-1	-	-

[1]The current session's attribute of the same name.

[2]As defined in the referenced **XDS** Specification).

[3] If two values are present, in either domestic or international communication, the syntax of the first shall be String (Printable), the syntax of the second String (Teletex), and the two shall convey the same information such that either can be safely ignored. Teletex strings allow inclusion, for example, of the accented characters commonly used in many countries. Not all input/output devices, however, permit the entry and display of such characters.

[4]For 1984 the syntax of the value shall be String (Printable).

[5]For 1984 at most one value shall be present.

[6]This attribute shall be present only if the ISDN Number attribute is present.

[7]For 1988 this attribute is required if any Organisational Unit Name is present.

ADMD Name

The name of the user's or DL's ADMD. It identifies the ADMD relative to the country that the Country Name attribute denotes. Its values are defined by that country.

Note: The attribute value comprising a single space is reserved. If permitted by the country that the Country Name attribute denotes, a single space designates any (that is, all) ADMDs within the country. This affects both the identification of users and DLs within the country and the routing of messages, probes and reports to and among the ADMDs of that country. Regarding the former, it requires that the O/R addresses of users and DLs within the country are chosen so as to ensure their unambiguousness, even in the absence of the actual names of the users' and DLs' ADMDs. Regarding the latter, it permits both PRMDs within, and ADMDs outside the country, to route messages, probes and reports to any of the ADMDs within the country indiscriminately, and requires that the ADMDs within the country interconnect themselves in such a way that the messages, probes and reports are conveyed to their destinations.

Common Name
The name commonly used to refer to the user or DL. It identifies the user or DL relative to the entity denoted by another attribute (for example, Organisation Name). Its values are defined by that entity.

Country Name
The name of the user's or DL's country. Its defined values are the numbers X.121 assigns to the country, or the character pairs ISO 3166 assigns to it.

Domain Type 1
The name of a class of information. Its values are defined by the user's or DL's ADMD and PRMD (if any) in combination.

Domain Type 2
The name of a class of information. Its values are defined by the user's or DL's ADMD and PRMD (if any) in combination.

Domain Type 3
The name of a class of information. Its values are defined by the user's or DL's ADMD and PRMD (if any) in combination.

Domain Type 4
The name of a class of information. Its values are defined by the user's or DL's ADMD and PRMD (if any) in combination.

Domain Value 1
An instance of the class of information that the Domain Type 1 attribute denotes.

Domain Value 2
An instance of the class of information that the Domain Type 2 attribute denotes.

Domain Value 3
An instance of the class of information that the Domain Type 3 attribute denotes.

Domain Value 4
An instance of the class of information that the Domain Type 4 attribute denotes.

Generation
The user's generation (for example, ''Jr'').

Given Name
The user's given name (for example, ''Robert'').

Initials
The initials of all of the user's names except his surname (for example, ''RE'').

ISDN Number
The ISDN number of the user's terminal. Its values are defined by E.163 and E.164.

ISDN Subaddress
The ISDN subaddress, if any, of the user's terminal. Its values are defined by E.163 and E.164.

Numeric User Identifier
Numerically identifies the user or DL relative to the ADMD that the ADMD Name attribute denotes. Its values are defined by that ADMD.

Organisation Name
The name of the user's or DL's organisation. As a national matter, such names may be assigned by the country that the Country Name attribute denotes, the ADMD that the ADMD Name attribute denotes, the PRMD that the PRMD Name attribute denotes, or the latter two organisations together.

Organisational Unit Name 1
The name of a unit (for example, division or department) of the organisation that the Organisation Name attribute denotes. The attribute's values are defined by that organisation.

Organisational Unit Name 2
The name of a subunit (for example, division or department) of the unit that the Organisational Unit Name 2 attribute denotes. The attribute's values are defined by the latter unit.

Organisational Unit Name 3
The name of a subunit (for example, division or department) of the unit that the Organisational Unit Name 2 attribute denotes. The attribute's values are defined by the latter unit.

Organisational Unit Name 4
The name of a subunit (for example, division or department) of the unit that the Organisational Unit Name 3 attribute denotes. The attribute's values are defined by the latter unit.

Postal Address Details
The means (for example, room and floor numbers in a large building) for identifying the exact point at which the user takes delivery of physical messages.

Postal Address in Full
The free-form and possibly multi-line postal address of the user, as a single teletex string, lines being separated as prescribed for teletex strings.

Postal Address in Lines
The free-form postal address of the user, in a sequence of printable strings, each representing a line of text.

Postal Code
The postal code for the geographical area in which the user takes delivery of physical messages. It identifies the area relative to the country that the Postal Country Name attribute denotes. Its values are defined by the postal administration of that country.

Postal Country Name
The name of the country in which the user takes delivery of physical messages. Its defined values are the numbers X.121 assigns to the country, or the character pairs ISO 3166 assigns to it.

Postal Delivery Point Name
Identifies the locus of distribution, other than that denoted by the Postal Office Name attribute (for example, a geographical area), of the user's physical messages.

Postal Delivery System Name
The name of the PDS through which the user is to receive physical messages. It identifies the PDS relative to the ADMD that the ADMD Name attribute denotes. Its values are defined by that ADMD.

Postal General Delivery Address
The code that the user gives to the post office that the Postal Office Name attribute denotes to collect the physical messages awaiting delivery to him. Its values are defined by that post office.

Postal Locale
Identifies the point of delivery, other than that denoted by the General Delivery Address, Postal Office Box Number, or Postal Street Address attribute (for example, a building or a hamlet), of the user's physical messages.

Postal Office Box Number
The number of the post office box by means of which the user takes delivery of physical messages. The box is located at the post office that the Postal Office Name attribute denotes. The attribute's values are defined by that post office.

Postal Office Name
The name of the municipality (for example, city or village) in which is situated the post office through which the user takes delivery of physical messages.

Postal Office Number
The means for distinguishing among several post offices denoted by the Postal Office Name attribute.

Postal Organisation Name
The name of the organisation through which the user takes delivery of physical messages.

Postal Patron Details
Additional information (for example, the name of the organisational unit through which the user takes delivery of physical messages) necessary to identify the user for purposes of physical delivery.

Postal Patron Name
The name under which the user takes delivery of physical messages.

Postal Street Address
The street address (for example, ''43 Primrose Lane'') at which the user takes delivery of physical messages.

Presentation Address
The presentation address of the user's terminal.

PRMD Name
The name of the user's PRMD. As a national matter, such names may be assigned by the country that the Country Name attribute denotes or the ADMD that the ADMD Name attribute denotes.

Surname
The user's surname (for example, ''Lee'').

Terminal Identifier
The terminal identifier of the user's terminal (for example, a Telex answer back or a Teletex terminal identifier).

Terminal Type
The type of the user's terminal. For its defined values, see Section 5.4.15 on page 150.

X121 Address

The network address of the user's terminal. Its values are defined by X.121.

Note: Among the strings admitted by X.121 are a telephone number preceded by the telephone escape digit (9), and a Telex number preceded by the Telex escape digit (8).

Certain attributes are grouped together for reference as follows. The Given Name, Initials, Surname and Generation attributes constitute the ''personal name attributes''. The Organisational Unit Name 1, Organisational Unit Name 2, Organisational Unit Name 3 and Organisational Unit Name 4 attributes constitute the *organisational unit name attributes*. The ISDN Number, ISDN Subaddress, Presentation Address and X121 Address attributes constitute the ''network address attributes''. For any *i* in the interval [1, 4], the Domain Type *i* and Domain Value *i* attributes constitute a domain-defined attribute (DDA).

Note: The widespread avoidance of DDAs produces more uniform and thus more user-friendly O/R addresses. However, it is anticipated that not all MDs will be able to avoid such attributes immediately. The purpose of DDAs is to permit an MD to retain its existing, native addressing conventions for a time. It is intended, however, that all MDs migrate away from the use of DDAs, and thus that DDAs are used only for an interim period.

An O/R address may take any of the forms summarised in the following table. The table indicates the attributes that may be present in an O/R address of each form. It also indicates whether it is mandatory (M) or conditional (C) that they do so. When applied to a group of attributes (for example, the network address attributes), ''mandatory'' means that at least one member of the group must be present, while ''conditional'' means that no members of the group need necessarily be present.

The presence or absence, in a particular O/R address, of conditional attributes is determined as follows. If a user or DL is accessed through a PRMD, the ADMD that the Country Name and ADMD Name attributes denote governs whether attributes used to route messages to the PRMD are present, but it imposes no other constraints on attributes. If a user or DL is *not* accessed through a PRMD, the same ADMD governs whether all conditional attributes except those specific to postal O/R addresses are present. All conditional attributes specific to postal O/R addresses are present or absent so as to satisfy the postal addressing requirements of the users they identify.

Attribute	Mnem[1]	Num[2]	Spost[3]	Upost[4]	Term[5]
ADMD Name	M	M	M	M	C
Common Name	C	-	-	-	-
Country Name	M	M	M	M	C
domain-defined attributes	C	C	-	-	C
network address attributes	-	-	-	-	M
Numeric User Identifier	-	M	-	-	-
Organisation Name	C	-	-	-	-
organ. unit name attributes	C	-	-	-	-
personal name attributes	C	-	-	-	-
Postal Address Details	-	-	C	-	-
Postal Address in Full	-	-	-	M	-
Postal Code	-	-	M	M	-
Postal Country Name	-	-	M	M	-
Postal Delivery Point Name	-	-	C	-	-
Postal Delivery System Name	-	-	C	C	-
Postal General Delivery Address	-	-	C	-	-
Postal Locale	-	-	C	-	-
Postal Office Box Number	-	-	C	-	-
Postal Office Name	-	-	C	-	-
Postal Office Number	-	-	C	-	-
Postal Organisation Name	-	-	C	-	-
Postal Patron Details	-	-	C	-	-
Postal Patron Name	-	-	C	-	-
Postal Street Address	-	-	C	-	-
PRMD Name	C	C[6]	C	C	C[6]
Terminal Identifier	-	-	-	-	C
Terminal Type	-	-	-	-	C

[1]Mnemonic. X.400 (1984) calls this Form 1 Variant 1.

[2]Numeric. X.400 (1984) calls this Form 1 Variant 2.

[3]Structured postal. For 1984 this O/R address form is undefined.

[4]Unstructured postal. For 1984 this O/R address form is undefined.

[5]X.400 (1984) calls this Form 1 Variant 3 and Form 2.

[6]For 1984 this attribute is absent (-). For 1988 it is conditional (C).

Table 5-32 Forms of O/R Address

Mnemonic O/R Address

Mnemonically identifies a user or DL. Using the ADMD Name and Country Name attributes, it identifies an ADMD. Using the Common Name attribute or the personal name attributes, the Organisation Name attribute, the organisational unit name attributes, the PRMD Name attribute, or a combination of these, and optionally DDAs, it identifies a user or DL relative to the ADMD.

The personal name attributes identify a user or DL relative to the entity denoted by another attribute (for example, Organisation Name). The Surname attribute shall be present if any of the other three personal name attributes are present.

Numeric O/R Address

Numerically identifies a user or DL. Using the ADMD Name and Country Name attributes, it identifies an ADMD. Using the Numeric User Identifier attribute and possibly the PRMD Name attribute, it identifies the user or DL relative to the ADMD. Any DDAs provide

information additional to that required to identify the user or DL.

Postal O/R Address

Identifies a user by means of its postal address. Two kinds of postal O/R address are distinguished:

Structured

Said of a postal O/R address that specifies a user's postal address by means of several attributes. The structure of the postal address is prescribed below in some detail.

Unstructured

Said of a postal O/R address that specifies a user's postal address in a single attribute. The structure of the postal address is left largely unspecified below.

Whether structured or unstructured, a postal O/R address does the following. Using the ADMD Name and Country Name attributes, it identifies an ADMD. Using the Postal Code and Postal Country Name attributes, it identifies the geographical region in which the user takes delivery of physical messages. Using the Postal Delivery System Name or PRMD Name attribute or both, it also may identify the PDS by means of which the user is to be accessed.

An unstructured postal O/R address also includes the Postal Address in Full attribute. A structured postal O/R address also includes every other postal addressing attribute that the PDS requires to identify the postal patron.

Note: The total number of characters in the values of all attributes but ADMD Name, Country Name and Postal Delivery System Name in a postal O/R address should be small enough to permit their rendition in 6 lines of 30 characters, the size of a typical physical envelope window. The rendition algorithm, while PDAU-defined, is likely to include inserting delimiters (for example, spaces) between some attribute values.

Terminal O/R Address

Identifies a user by identifying the user's terminal, using the network address attributes. It also may identify the ADMD through which the terminal is accessed, using the ADMD Name and Country Name attributes. The PRMD Name attribute and any DDAs, which shall be present only if the ADMD Name and Country Name attributes are, provide information additional to that required to identify the user.

If the terminal is a Telematic terminal, it gives the terminal's network address and possibly, using the Terminal Type and Terminal Identifier attributes, its terminal type and identifier. If the terminal is a Telex terminal, it gives the terminal's Telex number.

Whenever two O/R addresses are compared for equality, the following differences shall be ignored: (1) whether an attribute has a value whose syntax is String (Printable), a value whose syntax is String (Teletex), or both; (2) whether a letter in a value of an attribute not used in DDAs is upper- or lower-case; and (3) all leading, all trailing and all but one consecutive embedded spaces in an attribute value.

Notes:

1. An MD may impose additional equivalence rules upon the O/R addresses it assigns to its own users and DLs. It might define, for example, rules concerning punctuation characters in attribute values, the case of letters in attribute values, or the relative order of DDAs.
2. As a national matter, MDs may impose additional rules regarding any attribute that may have a value whose syntax is String (Printable), a value whose syntax is

String (Teletex), or both. In particular, the rules for deriving from a teletex string the equivalent printable string may be nationally prescribed.

5.3.32 OR Name

An instance of class **OR Name** comprises a directory name, an O/R address or both. The name is considered present if, and only if, the Directory Name attribute is present. The address comprises the attributes specific to the OR Address class and is considered present if, and only if, at least one of those attributes is present.

An O/R name's composition is context-sensitive. At submission, the name, the address, or both may be present. At transfer or delivery, the address is present and the name may (but need not) be present. Whether at submission, transfer or delivery, the MTS uses the name, if it is present, only if the address is absent or invalid.

The attributes specific to this class are listed in the following table.

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
Directory Name	Object (Name)[1]	-	0-1	-	1988

[1]As defined in the referenced **XDS** Specification).

Table 5-33 Attributes Specific to OR Name

Directory Name
The name assigned to the user or DL by the world-wide (that is, X.500) directory.

5.3.33 Per-recipient DR

An instance of class **Per-recipient DR** gives information about the successful delivery or the deliverability of the subject message to a particular recipient.

The attributes specific to this class are listed in the following table.

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
Delivery Point	Enum (Delivery Point)	-	1	private-UA	-
Delivery Time	String (UTC Time)	0-17	1	-	-
Proof of Delivery	Object (Algorithm and Result)	-	0-1	-	1988
Recipient Certificate	Object (Certificates)[1]	-	0-1	-	1988

[1]As defined in the referenced **XDS** Specification).

Table 5-34 Attributes Specific to Per-recipient DR

Delivery Point
The nature of the functional entity by means of which the subject message was or would have been delivered to the recipient. For its defined values, see Section 5.4.4 on page 144.

Delivery Time
The date and time at which the subject message was or would have been delivered to the recipient.

Proof of Delivery
Proof that the message has been delivered to the recipient. It is present if the originator requested proof of delivery. The algorithm involved is applied to an instance of the Proof of Delivery Basis class.

Recipient Certificate
The recipient's certificate. Generated by a trusted source (for example, a CA), it constitutes a verified copy of the recipient's PAEK. It is present if the originator requested proof of delivery and an asymmetric encryption algorithm was used to compute the proof.

5.3.34 Per-recipient NDR

An instance of class **Per-recipient NDR** gives information about the unsuccessful delivery or the undeliverability of the subject message to a particular recipient.

The attributes specific to this class are listed in the following table.

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
Non-delivery Diagnostic	Enum (Diagnostic)	-	1	no-diagnostic	-
Non-delivery Reason	Enum (Reason)	-	1	transfer-failed	-

Table 5-35 Attributes Specific to Per-recipient NDR

Non-delivery Diagnostic
Why in detail the subject message was not, or would not, have been conveyed to the recipient. For its defined values, see Section 5.4.5 on page 144.

Non-delivery Reason
Identifies the factor that prevented, or would have prevented, the subject message from being conveyed to the recipient. For its defined values, see Section 5.4.10 on page 148.

5.3.35 Per-recipient Report

An instance of class *Per-recipient Report* gives information about the successful or unsuccessful delivery, or the deliverability or undeliverability, of the subject message to a particular recipient.

The attributes specific to this class are listed in the following table.

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
Actual Recipient Name	Object (OR Name)	-	1	-	-
Arrival Time	String (UTC Time)	0-17	1	-	-
Converted EITs	Object (EITs)	-	0-1	-	-
Forwarding Address	Object (OR Name)	-	0-1	-	1988
Intended Recipient Number	Integer	-	1	-	-
Originally Intended Recip	Object (OR Name)	-	0-1	-	-
Originator Report Request	Enum (Report Request[1])	-	1	non-delivery	-
Redirection History	Object (Redirection Record)	-	0-512	-	1988
Supplementary Info	String (Printable)	1-256[2]	0-1	-	-

[1]The value shall not be **always-audited**.

[2]For 1984 the maximum value length is 64. For 1988 it is 256.

Table 5-36 Attributes Specific to Per-recipient Report

Actual Recipient Name
The O/R name of the recipient to which the per-recipient report pertains. If the report concerns a message (not a probe) and the recipient is an alternate recipient, this attribute's value is the O/R name of that alternate recipient.

Arrival Time
The date and time at which the communique entered the MD that produced the per-recipient report.

Converted EITs
The EITs that characterise, or would characterise, the subject message after its conversion. This attribute is present if, and only if, the MTS converted, or would have converted, the subject message.

Forwarding Address
The new postal O/R address of the recipient, a PDS patron. It is present only if the originator requested the recipient's physical forwarding address.

Intended Recipient Number
The ordinal position of the intended recipient in the list of recipients specified by the communique's originator. For the first such recipient this attribute's value is one.

Originally Intended Recip
The O/R name of the originally intended recipient. This attribute is present if, and only if, the Actual Recipient Name attribute denotes an alternate recipient.

Originator Report Request
The kinds of report that the originating user requested in the circumstances prescribed for such reports. For its defined values, see Section 5.4.13 on page 149.

Redirection History
Documents the communique's redirection. The redirection records appear in chronological order.

Supplementary Info
Supplementary information concerning the fate of the communique. This attribute is present at the option of the MD that produces the per-recipient report.

Note: This attribute may be used, for example, by a Teletex AU or Teletex/Telex conversion facility. In such cases, it may provide the received answer-back, the Telex transmission duration, or the note and received recorded message.

5.3.36 Probe

An instance of class **Probe** is a secondary information object conveyed between users by the MTS. It tests the deliverability to prescribed users of a submitted message having prescribed characteristics.

The attributes specific to this class are listed in the following table.

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
Content Length	Integer	-	0-1	-	-

Table 5-37 Attributes Specific to Probe

Content Length
The length in octets of the Content attribute of the message whose deliverability the probe is intended to test. This attribute is present at the option of the UA that submits the probe.

5.3.37 Probe RD

An instance of class **Probe RD** identifies an intended recipient of a probe (or message) and records certain information about that recipient.

The attributes specific to this class are listed in the following table.

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
MTA Report Request	Enum (Report Request[1])	-	1	non-delivery	-
MTA Responsibility	Boolean	-	1	true	-
Recipient Number	Integer	-	1	-	-
Redirection History	Object (Redirection Record)	-	0-512	-	1988

[1]The value shall not be **never**.

Table 5-38 Attributes Specific to Probe RD

MTA Report Request
The kinds of report that the originating MD requests in the circumstances prescribed for such reports. For its defined values, see Section 5.4.13 on page 149.

This attribute shall neither exclude reports nor call for less external trace information than the Originator Report Request attribute does.

MTA Responsibility
Whether the probe (or message), of which the RD is a part, is to be delivered to the recipient.

Note: If this attribute is **false**, another copy of the probe (or message) has been created and is being independently routed to this particular recipient. The present copy shall not be routed for eventual delivery to the recipient.

Recipient Number
The recipient's ordinal position in the list of recipients originally specified by the originator of the probe (or message) of which the recipient specifier is a part. For the first recipient this attribute's value is one.

Redirection History
Documents the probe's redirection in the course of its conveyance to the recipient. The redirection records appear in chronological order.

5.3.38 RD

An instance of class *RD* identifies an intended recipient of a communique and records certain information about that recipient.

The attributes specific to this class are none.

5.3.39 Redirection Record

An instance of class **Redirection Record** documents a communique's redirection.

The attributes specific to this class are listed in the following table.

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
Intended Recipient Name	Object (OR Name)	-	1	-	1988
Reason	Enum (Redirection Reason)	-	1	-	1988
Time	String (UTC Time)	0-17	1	-	1988

Table 5-39 Attributes Specific to Redirection Record

Intended Recipient Name
The O/R name of the recipient to which the communique was being conveyed when it was redirected.

Reason
Indicates why the communique was redirected. For its defined values, see Section 5.4.11 on page 149.

Time
The date and time at which redirection occurred.

5.3.40 Report

An instance of class **Report** is a secondary information object conveyed by the MTS between users. It reports the successful or unsuccessful delivery, or the deliverability or undeliverability, of the subject message to some or all of its recipients. A single report may report both successful and unsuccessful delivery, or both deliverability and undeliverability (to different recipients).

The attributes specific to this class are listed in the following table.

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
External Trace Info	Object (External Trace Entry)	-	0-512[2]	-	-
Internal Trace Info	Object (Internal Trace Entry)	-	0-512	-	-
MTS Identifier	Object (MTS Identifier)	-	1	*see note*[1]	-
Per-recipient Reports	Object (Per-recipient Report)	-	1-32767	-	-
Report Additional Info	String (Encoding)	3-1024	0-1	-	-
Report Destination	Object (OR Name)	-	1	-	-
Subject External Trace Info	Object (External Trace Entry)	-	0-512	-	-

[1]As assigned by the service to identify the report as prescribed by X.400.

[2]The receiving application will receive at least one. If client does not provide an external trace entry, the service must do so.

Table 5-40 Attributes Specific to Report

External Trace Info

External trace entries which document how the report was acted upon by each and every MD that transferred it. The positional order of attribute values reflects the time order of MD processing, the first value being produced by the MD that originated the report.

A single MD may add one or several trace entries to the attribute as follows. If it simply transfers the report to the MD that is its first choice, it adds to the attribute a single trace entry whose Action attribute has the value **relayed**. Otherwise it adds one or more trace entries whose Action attributes have the value **rerouted**; these several trace entries may indicate attempts to transfer the report to one or several MDs.

Note: This attribute provides, among other things, a basis for loop detection.

Internal Trace Info

Internal trace entries which document how the report was acted upon by each and every MTA that transferred it. However, at the service's option (and in common practice), the scope of this attribute may be limited to the local MD. The positional order of attribute values reflects the time order of MTA processing.

A single MTA may add one or several trace entries to the attribute as follows. If it simply transfers the report to the MTA that is its first choice, it adds to the attribute a single trace entry whose Action attribute has the value **relayed**. Otherwise it adds one or more trace entries whose Action attributes have the value **rerouted**; these several trace entries may indicate attempts to transfer the report to one or several MTAs.

Note: This attribute provides, among other things, a basis for loop detection. Note also that this attribute is defined by MOTIS and X.400 (1988) but not by X.400 (1984).

MTS Identifier

The MTS identifier of the report.

Per-recipient Reports

Delivered per-recipient reports, one for each recipient to which the report applies.

Report Additional Info

Binary data that follows the BER. This attribute is present by bilateral agreement between the MD that originates the report and the MD serving the originator of the communique. The data type of the data value that the attribute value encodes is part of the same agreement, and is thus presumed known by both MDs a priori.

Report Destination

The O/R name of the report's immediate destination. The MTA that originates the report initialises this attribute to the last O/R name in the Expansion History attribute if it is present in the communique, and to the O/R name of the communique's originator otherwise. A DL expansion point may replace its own O/R-name with either the O/R name that immediately precedes it in the report's Expansion History, or some other O/R name, according to the DL's reporting policy.

Subject External Trace Info

External trace entries which document how the communique was acted upon by each and every MD that transferred it. The External Trace Info of the communique at the time it entered the MD that produced the report. This attribute is present if, and only if, among the values of the Per-recipient Reports attribute (specific to the Delivered Report class), there is a per-recipient report for an intended recipient, the MTA Report Request attribute of whose RD is **always-audited**.

5.3.41 Security Label

An instance of class **Security Label** is a security-related descriptor which may be assigned to a communique or a report, or to any of the functional entities which convey communiques and reports within the MHS. The security policy in force prescribes how security labels are assigned and used.

Security labels may be assigned to communiques, reports, UAs, MSs, MTAs, MDs, associations between a UA or an MS and an MTA or an MD, and associations between two MTAs or MDs. Security labels may be assigned to other functional entities within the MTS (for example, secure routes) as a local matter or by bilateral agreement.

When security labels are assigned to UAs, MSs, MTAs and MDs, their handling of communiques and reports bearing security labels is governed by the labels and the security context in affect at submission, transfer or delivery, in line with the security policy in force. If security labels are not so assigned, the assignment of labels to communiques at submission, the transfer of communiques and reports, and the delivery of messages and reports is discretionary. An MTA may hold for later delivery messages and reports not immediately deliverable for security reasons.

The attributes specific to this class are listed in the following table.

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
Privacy Mark	String (Printable)	1-128	0-1	-	1988
Security Category Data	*any*	-	0-64	-	1988
Security Category IDs	String (Object Identifier)	-	0-64	-	1988
Security Classification	Enum (Security Classification)	-	1	unmarked	1988
Security Policy ID	String (Object Identifier)	-	0-1	-	1988

Table 5-41 Attributes Specific to Security Label

Privacy Mark

Identifies the level of privacy to be afforded a communique or report. Its values (for example, ''IN CONFIDENCE'' and ''IN STRICTEST CONFIDENCE'') may (but need not) be defined by the associated security policy. The presence or absence of this attribute is determined by the security policy in force.

Security Category Data
Parameters associated with the values of the Security Category IDs attribute. The two attributes shall have the same number of values, their *i*th values corresponding to one another.

Security Category IDs
Designators (for example, ''PERSONAL'', ''STAFF'' and ''COMMERCIAL'') for restrictions within the context of the security classification, the privacy mark, or both. The values may be defined by the associated security policy or bilaterally agreed. The presence or absence of this attribute is determined by the security policy in force.

Security Classification
Classifies a communique or report for security purposes. For its defined values, see Section 5.4.14 on page 150. Classification.

Security Policy ID
Identifies the security policy with which the security label is associated. The presence or absence of this attribute is determined by the security policy in force.

5.3.42 Session

An instance of class **Session** is an MA or MT session between the client and the service. Among other things, a session comprises information about an MTA's environment, especially information about the ''subject domain'', the MD that contains the MTA. The ''local environment'' is the environment of the local MTA. The ''local MTA'' is the MTA that comprises the client and the service. The local MTA is part of the ''local MD''.

The attributes specific to this class are listed in the following table.

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
ADMD Name	String (Printable)	0-16	1	*see note*[1]	-
Country Name	String (Printable)	2-3	1	*see note*[1]	-
Event Handle	Integer	-	0-1	-	-
PRMD Identifier	String (Printable)	1-16	0-1	*see note*[1]	-

[1]That of the local environment.

Table 5-42 Attributes Specific to Session

ADMD Name
The name of the subject domain, if an ADMD, or of the ADMD to which the subject domain is attached, if a PRMD. It identifies the ADMD relative to the country that the Country Name attribute denotes. Its values are defined by that country.

Country Name
The name of the country of the ADMD that the ADMD Name attribute denotes. Its defined values are the numbers X.121 assigns to the country, or the character pairs ISO 3166 assigns to it.

Event Handle
If present, an event handle that may be supplied to a system-defined function or operating system call to achieve the effect of invoking the *Wait*( ) function. This system-defined primitive may be more capable than the *Wait*( ) function; in particular, when properly invoked, it may block the client until the first of several events occurs.

PRMD Identifier
The identifier of the subject domain, if a PRMD. It identifies the PRMD relative to the ADMD that the ADMD Name attribute denotes. Its values are defined by that ADMD. This attribute is present if, and only if, the subject domain is a PRMD.

Note: The defined values of this attribute may (but need not) be identical to those of the PRMD Name attribute (if any) of an O/R name of a user served by the subject domain.

5.3.43 Submission Results

An instance of class **Submission Results** is an acknowledgement of the successful submission to the MTS of a submitted communique. The MTS provides the acknowledgement.

The attributes specific to this class are listed in the following table.

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
Content Identifier	String (Printable)	1-16	0-1	-	-
MTA Certificate	Object (Certificates)[1]	-	0-1	-	1988
MTS Identifier	Object (MTS Identifier)	-	1	-	-
Proof of Submission	Object (Algorithm and Result)	-	0-1	-	1988
Submission Time	String (UTC Time)	0-17	1	-	-

[1]As defined in the referenced **XDS** Specification.

Table 5-43 Attributes Specific to Submission Results

Content Identifier
Information facilitating the correlation with the communique of any reports it may provoke. The like-named attribute of the submitted communique.

MTA Certificate
The certificate of the MTA to which the message was submitted. Generated by a trusted source (for example, a CA), it constitutes a verified copy of the MTA's PAEK. It is present if the originator requested proof of submission and an asymmetric encryption algorithm is used to compute the proof.

MTS Identifier
The MTS identifier assigned to the submitted communique.

Proof of Submission
Proof that the message has been submitted to the MTS. It is present if the originator requested it. The algorithm involved is applied to an instance of the Proof of Submission Basis class.

Submission Time
The date and time at which the MTS considers the submitted communique to have been submitted.

5.3.44 Submitted Communique

An instance of class *Submitted Communique* is a primary information object submitted by users to the MTS.

The attributes specific to this class are listed in the following table.

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
Alternate Recipient Allowed	Boolean	-	1	false	-
Content Correlator	*any*	3-512	0-1	-	1988
Content Identifier	String (Printable)	1-16	0-1	-	-
Content Type	Integer or String (Object Identifier)[1]	-	1	P2-1984	-
Conversion Loss Prohibited	Boolean	-	1	false	1988
Conversion Prohibited	Boolean	-	1	false	-
Expansion Prohibited	Boolean	-	1	false	1988
Origin Check	Object (Algorithm and Result)	-	0-1	-	1988
Original EITs	Object (EITs)	-	0-1	-	-
Originator Certificate	Object (Certificates)[2]	-	0-1	-	1988
Originator Name	Object (OR Name)	-	1	-	-
Reassignment Prohibited	Boolean	-	1	false	1988
Recipient Descriptors	Object (RD)	-	1-32767	-	-
Security Label	Object (Security Label)	-	0-1	-	1988

[1]For 1984 the syntax is Integer. For 1988 it is Integer or String (Object Identifier).

[2]As defined in the referenced **XDS** Specification.

Table 5-44 Attributes Specific to Submitted Communique

Alternate Recipient Allowed
Whether the originator permits the MTS to deliver the subject message to an alternate recipient. An MD may (but need not) assign a user, the alternate recipient, to accept delivery of messages whose Recipient Descriptors attributes contain O/R names that are invalid but recognised as meant to denote users of that MD.

Content Correlator
Information facilitating the correlation with the submitted communique of any reports it may provoke. This attribute is present at the option of the originator's UA. It is not conveyed to recipients at delivery.

Content Identifier
Information facilitating the correlation with the submitted communique of any reports it may provoke. This attribute is present at the option of the originator's UA. It is conveyed to recipients at delivery.

Content Type
Identifies the syntax and semantics of the value of the Content attribute of the subject message. Its defined values are as prescribed for the like-named attribute specific to the Communique class.

Conversion Loss Prohibited
Whether the originator prohibits the MTS from converting the subject message (should such conversion be necessary) if it would cause loss of information as defined in X.408.

Conversion Prohibited
Whether the originator prohibits the MTS from converting the subject message (should such

conversion be necessary) under any circumstances.

Expansion Prohibited
Whether the originator instructs the MTS to issue an NDR rather than expand a DL if the O/R name specified for any of the recipients proves to denote a DL not a user.

Origin Check
A means by which a third party (for example, a user or an MTA) can verify the submitted communique's origin. This attribute is present at the option of the originator's UA. The algorithm involved is applied to an instance of the Origin Check Basis class.

Original EITs
The EITs of the Content attribute of the subject message. This attribute is present at the option of the originator's UA.

Originator Certificate
The originator's certificate. Generated by a trusted source (for example, a CA), it constitutes a verified copy of the originator's PAEK. This attribute is present at the option of the originator's UA.

Originator Name
The O/R name of the submitted communique's originator.

Reassignment Prohibited
Whether the originator prohibits the intended recipients from redirecting the submitted communique.

Recipient Descriptors
The RDs of the submitted communique's intended recipients.

Security Label
The security label associated with the submitted communique. It shall be assigned in line with the security policy in force.

5.3.45 Submitted Message

An instance of class **Submitted Message** is a primary information object submitted by users to the MTS. It conveys arbitrary binary data from one user, the originator, to one or more users, the recipients.

The attributes specific to this class are listed in the following table.

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
Confidentiality Algorithm	Object (Algorithm)	-	0-1	-	1988
Content	Object (Content)	-	1	-	-
Content Return Requested	Boolean	-	1	false	-
Deferred Delivery Time	String (UTC Time)	0-17	0-1	-	-
Disclosure Allowed	Boolean	-	1	false	-
Latest Delivery Time	String (UTC Time)	0-17	0-1	-	1988
Originator Return Address	Object (OR Address)	-	0-1	-	1988
Priority	Enum (Priority)	-	1	normal	-
Proof of Submission Requested	Boolean	-	1	false	1988

Table 5-45 Attributes Specific to Submitted Message

Confidentiality Algorithm
Identifies the algorithm that the originator of the submitted message used to encrypt its content and which the recipients may use to decrypt it.

The algorithm may be either symmetric or asymmetric. If the former, the associated key may be derived from the Token attribute of any of the submitted message's RDs or, alternatively, distributed by some other means. If the latter, the originator may use the intended recipient's public key to encrypt the content, and the recipient may use the associated secret key to decrypt it. The submitted message must be addressed to either a single recipient or a group of recipients sharing the same key pair.

Content The arbitrary binary information the submitted message is intended to convey to its recipients. The MTS modifies this attribute's value only for purposes of conversion.

Content Return Requested
Whether the Content attribute is to be included as the like-named attribute of any NDRs the submitted message provokes.

Deferred Delivery Time
The date and time, if any, before which the submitted message shall not be delivered. Delivery deferral is normally the responsibility of the MD that originates the submitted message. Thus messages whose Deferred Delivery Time attributes are present shall be transferred between MDs only by bilateral agreement between those MDs.

Disclosure Allowed
Whether the O/R names of other recipients are to be indicated to each recipient at delivery.

Latest Delivery Time
The date and time after which the MTS is to treat the submitted message as undeliverable if it has not yet been delivered to a particular recipient.

Originator Return Address
The postal O/R address of the submitted message's originator. It shall be present if the originator supplied a postal O/R address for an intended recipient or included physical delivery among a recipient's preferred delivery modes. It may also be present if a recipient DL contains, or is likely to contain, one or more members for whom physical delivery is required.

Priority
The relative priority at which the submitted message is to be transferred. For its defined values, see Section 5.4.9 on page 148.

Proof of Submission Requested
Whether the originator of the submitted message requires proof of its submission.

5.3.46 Submitted Message RD

An instance of class **Submitted Message RD** identifies an intended recipient of a submitted message and records certain information about that recipient.

The attributes specific to this class are listed in the following table.

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
Bureau FAX Delivery	Boolean	-	1	false	1988
Forwarding Address Requested	Boolean	-	1	false	1988
Forwarding Prohibited	Boolean	-	1	false	1988
Integrity Check	Object (Algorithm and Result)	-	0-1	-	1988
Postal Mode	Enum (Postal Mode)	-	0-1	-	1988
Postal Report	Enum (Postal Report)	-	0-1	-	1988
Proof of Delivery Requested	Boolean	-	1	false	1988
Recipient Number for Advice	String (Teletex)	1-32	0-1	-	1988
Registration	Enum (Registration)	-	0-1	-	1988
Token	Object (Token)	-	0-1	-	1988

Table 5-46 Attributes Specific to Submitted Message RD

Bureau FAX Delivery

Whether the message is to be delivered to the recipient by Bureau FAX. This attribute shall be present only if the originator supplied a postal O/R address for the intended recipient or included physical delivery among the recipient's preferred delivery modes.

Note: Bureau FAX delivery comprises all A-H modes of delivery defined in F.170: regular delivery (A), special delivery (B), express mail (C), counter collection (D), counter collection with telephone advice (E), telefax (F), counter collection with Telex advice (G), and counter collection with Teletex advice (H).

Forwarding Address Requested

Whether the recipient's physical forwarding address is to be returned in an NDR. It may be **true** if the originator supplied a postal O/R address for the intended recipient or included physical delivery among the recipient's preferred delivery modes. (It may be **true** even if physical forwarding is prohibited.)

Forwarding Prohibited

Whether physical forwarding of the message is prohibited for the recipient. It may be **true** if the originator supplied a postal O/R address for the intended recipient or included physical delivery among the recipient's preferred delivery modes.

Integrity Check

A means by which the recipient can verify that the value of the message's Content attribute is unchanged. The algorithm involved is applied to an instance of the Integrity Check Basis class.

Postal Mode

Identifies the kind of physical delivery to be employed for the recipient. For its defined values, see Section 5.4.7 on page 147.

This attribute shall be present only if the originator supplied a postal O/R address for the intended recipient or included physical delivery among the recipient's preferred delivery modes. If no value is present, the value **ordinary-mail** is implied.

Postal Report

Identifies the kind of physical delivery report to be issued for the recipient. For its defined values, see Section 5.4.8 on page 148.

This attribute shall be present only if the originator supplied a postal O/R address for the intended recipient or included physical delivery among the recipient's preferred delivery modes. If no value is present the value **undeliverable-mail-via-pds** is implied.

Proof of Delivery Requested
Whether the originator of the message requires proof of its delivery to the recipient.

Recipient Number for Advice
The recipient's telephone, Telex or Teletex number. It shall be present if the originator supplied a postal O/R address for the intended recipient or included physical delivery among the recipient's preferred delivery modes, and the Postal Mode argument specifies **cc-with-advice** or the Bureau FAX Delivery attribute is **true**.

Registration
Identifies the kind of registered mail service to be employed in the message's physical delivery to the recipient. For its defined values, see Section 5.4.12 on page 149.

This attribute shall be present only if the originator supplied a postal O/R address for the intended recipient or included physical delivery among the recipient's preferred delivery modes. If no value is present the value **unregistered-mail** is implied.

Token
The token prepared for the recipient. The algorithm involved is applied to an instance of the Asymmetric Token class.

5.3.47 Submitted Probe

An instance of class **Submitted Probe** is a secondary information object submitted by users to the MTS. It tests the deliverability to prescribed users of a submitted message having prescribed characteristics.

The attributes specific to this class are listed in the following table.

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
Content Length	Integer	-	0-1	-	-

Table 5-47 Attributes Specific to Submitted Probe

Content Length
The length in octets of the Content attribute of the message whose deliverability the probe is intended to test. This attribute is present at the option of the UA that submits the probe.

5.3.48 Submitted Probe RD

An instance of class **Submitted Probe RD** identifies an intended recipient of a submitted probe (or message) and records certain information about that recipient.

The attributes specific to this class are listed in the following table.

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
Alternate Recipient Name	Object (OR Name)	-	0-1	-	1988
Explicit Conversion	Enum (Explicit Conversion)	-	1	no-conversion	-
Originator Report Request	Enum (Report Request[1])	-	1	non-delivery	-
Preferred Delivery Modes	Enum (Delivery Mode)	-	1-10	any	1988
Recipient Name	Object (OR Name)	-	1	-	-
Rendition Attributes	String (Object Identifier)	-	0-1	-	1988

[1]The value shall not be **always-audited**.

Table 5-48 Attributes Specific to Submitted Probe RD

Alternate Recipient Name
The O/R name of the alternate recipient requested by the originator of the probe (or message). The MTS redirects the probe (or message) to that user or DL if it cannot establish its deliverability to the recipient specified by the originator, added as a result of DL expansion, or substituted as a result of recipient redirection. The probe (or message) is considered deliverable to the originator's alternate recipient in preference to that assigned by the recipient's MD.

Explicit Conversion
The type of conversion that the originator requests is performed for the recipient. For its defined values, see Section 5.4.6 on page 147.

Originator Report Request
The kinds of report that the originator requests in the circumstances prescribed for such reports. For its defined values, see Section 5.4.13 on page 149.

Preferred Delivery Modes
Identifies the modes of delivery requested by the originator, in order of decreasing preference. It indicates, that is, which of the O/R addresses in the recipient's directory entry should be employed for the purpose of message delivery. For its defined values, see Section 5.4.3 on page 143.

This attribute is relevant only if the recipient's O/R name contains a directory name but not an O/R address. If the entry contains no O/R address consistent with any of the specified delivery modes, the service refuses submission, reporting an exception, or transfer, issuing an NDR.

These preferences override any the recipient may have expressed (for example, in its directory entry), even to the extent that EIT incompatibilities that cannot be resolved by conversion provoke an NDR.

Recipient Name
The recipient's O/R name.

Rendition Attributes
Identifies the message's rendition attributes. It may be present if the originator supplied a postal O/R address for the intended recipient or included physical delivery among the recipient's preferred delivery modes.

Among the attribute's defined values is **basic-rendition**, meaning a PDAU's basic or default rendition and denoting the object identifier specified in ASN.1 as *{joint-iso-ccitt mhs-motis(6) mts(3) 5 0}*. MDs may use other values by bilateral agreement.

5.3.49 Teletex NBPs

An instance of class **Teletex NBPs** comprises the NBPs of a Teletex document. Each NBP identifies a non-basic capability of a Teletex terminal.

Note: In the Teletex service, NBPs are conveyed between terminals by means of the Command Document Start (CDS) data structure of T.62.

The attributes specific to this class are listed in the following table.

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
Control Character Sets	String (Teletex)	see note[1]	0-1	-	-
Graphic Character Sets	String (Teletex)	see note[1]	0-1	-	-
Miscellaneous Capabilities	String (Teletex)	see note[1]	0-1	-	-
Page Formats	String (Octet)	see note[1]	0-1	-	-
Private Use	String (Octet)	0-126	0-1	-	-

[1]As prescribed for Teletex.

Table 5-49 Attributes Specific to Teletex NBPs

Control Character Sets
The information that signals, in a CDS, the like-named non-basic Teletex terminal capability. This attribute is present at the client's option.

Graphic Character Sets
See Control Character Sets.

Miscellaneous Capabilities
See Control Character Sets.

Page Formats
See Control Character Sets.

Private Use
See Control Character Sets.

5.3.50 Token

An instance of class **Token** gives to the recepient of the token protected security-relevant information.

The attributes of this class are none.

5.3.51 Token Public Data

This is an abstract class, providing for future expansion of other token types.

The attributes specific to this class are none.

5.4 Syntax Definitions

This section defines the MH enumeration syntaxes, that is, the syntaxes in the Enumeration group that are specific to MH.

5.4.1 Action

An instance of enumeration syntax `Action` identifies the action that an MD or MTA may take with respect to a communique or report.

For 1984 its value is chosen from the following:

relayed
The MD or MTA transfers the communique or report to another MD or MTA, respectively (in the normal way).

rerouted
The MD or MTA attempts but fails to transfer the communique or report to another MD or MTA, respectively (an exceptional condition).

recip reassigned
This value is a MOTIS ISO/DIS 8883 extension to X.400 1984.

If interworking with an A/311 or NIST OIW conformant 1988 MTA then this value (in combination with an immediately preceeding trace element) can be equivalent to an 1988 action of **expanded** or **redirected**. This transformation is performed as part of internal-trace interworking between 1988 and 1984 MOTIS extended MTAs. (See documents referenced above.)

For 1988 its value may be chosen from **relayed**, **rerouted**, and additionally from the following:

expanded
The MD or MTA expands a DL.

redirected
The MD or MTA redirects the communique (but not the report).

5.4.2 Builtin EIT

An instance of enumeration syntax `Builtin EIT` identifies one of the kinds of information that may be in a message's content. Its value is chosen from those in the following table. The first column of the table lists the symbolic values, the second their meanings, and the third column the documents that specify the EITs the values denote.

Value	Meaning	References
g3-fax	G3 facsimile images	T.4, T.30
g4-class-1	G4 Class 1 facsimile final-form documents	T.5, T.6, T.400, T.503
ia5-text	International Alphabet No. 5 (IA5) text	T.50
iso-6937-text	ISO 6937 text[1]	ISO 6937
mixed-mode	Mixed-mode Teletex & G4 Classes 2-3 facsimile final-form documents	T.400, T.501
oda	Office Document Architecture[1]	T.411
teletex	Teletex documents	F.200, T.61, T.60
telex	International Alphabet No. 2 (ITA2) text	F.1
undefined	Information whose kind is not listed in this table	-
videotex	Videotex data	T.100, T.101

[1]This EIT is not defined by X.400, but is defined by various functional standards.

Table 5-50 Values of the Enumeration (Builtin EIT) Syntax

Note: The interfaces do not support X.400's Simple Formattable Document (SFD) and Voice EITs. X.400 (1984) does not fully specify the Voice type. X.400 (1988) abandons the SFD type, and leaves the Voice type only partially specified.

5.4.3 Delivery Mode

An instance of enumeration syntax `Delivery Mode` identifies a mode of message or report delivery. Its value is chosen from the following:

any
Any of the following delivery modes.

g3-fax
Delivery to a G3 facsimile terminal.

g4-fax
Delivery to a G4 facsimile terminal.

ia5-terminal
Delivery to an IA5 terminal.

mts
Delivery via the MTS.

pds
Delivery via a physical delivery system (PDS) (for example, the postal system).

telephone
Delivery via telephone.

teletex
Delivery to a Teletex terminal.

telex
Delivery to a Telex terminal.

videotex
Delivery to a Videotex terminal.

5.4.4 Delivery Point

An instance of enumeration syntax `Delivery Point` identifies the nature of the functional entity by means of which a message may be delivered to a recipient.

For 1984 its value is chosen from the following:

private-UA
A UA owned by an organisation other than an Administration (for example, a UA that is part of an in-house messaging system).

public-UA
A UA owned by an Administration (that is, a public service provider).

For 1988 its value may be chosen, additionally, from the following:

dl
A distribution list (DL).

ms
A message store (MS).

other-au
An access unit (AU) other than a PDAU.

pdau
A physical delivery access unit (PDAU).

pds-patron
A physical delivery system (PDS) patron.

5.4.5 Diagnostic

An instance of enumeration syntax `Diagnostic` explains in detail why a message cannot be delivered to a recipient.

For 1984 its value is chosen from the following:

content-too-long
The recipient does not accept delivery of messages whose Content attribute is as long as that of the message.

conversion-unsubscribed
The recipient does not subscribe to a required conversion.

eits-unsupported
The recipient does not accept delivery of messages having the message's EITs.

impractical-to-convert
A required conversion is impractical.

loop-detected
The message looped within the MTS.

maximum-time-expired
The message cannot be delivered in the time allowed.

mts-congested
The MTS was congested.

no-diagnostic
No diagnostic information is supplied.

or-name-ambiguous
The recipient's O/R name designates several users.

or-name-unrecognised
The recipient's O/R name is unrecognised.

parameters-invalid
The message is malformed.

prohibited-to-convert
The originator prohibited a required conversion.

recipient-unavailable
The recipient is unavailable.

For 1988 its value may be chosen, additionally, from the following:

alphabetic-character-lost
A conversion essential to the message's delivery would cause one or more alphabetic characters to be lost.

content-syntax-in-error
The message's content is syntactically in error.

content-type-unsupported
The MTS cannot process the content of the message in a manner essential to its delivery because the MTS does not support contents of the message's type.

conversion-loss-prohibited
A conversion essential to the message's delivery would cause loss of information and the originator prohibited such loss.

critical-function-unsupported
An MTA does not support a function critical to the message's transfer or delivery.

downgrading-impossible
The message cannot be transferred because it cannot be downgraded (see Annex B of **Recommendation X.419**, in Referenced Documents).

expansion-failed
The MTS cannot complete the expansion of a DL.

expansion-prohibited
The originator prohibited the expansion of DLs.

length-constraint-violated
A value of an attribute of the message, or of one of its subobjects, violates the attribute's value length constraints and the MTS cannot handle the value.

line-too-long
A conversion essential to the message's delivery would cause loss of information because a line is too long.

mail-address-incomplete
The recipient's postal O/R name is incomplete.

mail-address-incorrect
The recipient's postal O/R name is incorrect.

mail-forwarding-prohibited
Physical forwarding is required and the originator has prohibited it.

mail-forwarding-unwanted
Physical forwarding is required and the recipient does not desire it.

mail-new-address-unknown
The recipient's postal O/R name identifies a postal patron who has moved and whose new address is unknown.

mail-office-incorrect-or-invalid
The recipient's postal O/R name attempts to but does not correctly identify a physical delivery office.

mail-organisation-expired
The recipient's postal O/R name identifies an organisation that has expired.

mail-recipient-deceased
The recipient's postal O/R name identifies someone who is deceased.

mail-recipient-departed
The recipient's postal O/R name identifies someone who has changed temporary address (that is, departed) and to whom forwarding does not apply.

mail-recipient-moved
The recipient's postal O/R name identifies someone who has changed address permanently (that is, moved) and to whom forwarding does not apply.

mail-recipient-travelling
The recipient's postal O/R name identifies someone who has changed address temporarily (that is, is travelling) and to whom forwarding does not apply.

mail-recipient-unknown
The recipient's postal O/R name identifies someone who is unknown at that address.

mail-refused
The recipient's postal O/R name identifies someone who refuses to accept delivery of the message.

mail-unclaimed
The recipient's postal O/R name identifies someone who did not collect the message.

multiple-information-losses
A conversion essential to the message's delivery would result in loss of information of several kinds.

no-bilateral-agreement
The message's delivery requires a bilateral agreement but no such agreement exists.

number-constraint-violated
An attribute of the message, or of one of its components, violates the attribute's value number constraints in that at least one value of the attribute is required but none is present.

page-too-long
A conversion essential to the message's delivery would cause loss of information because a

page is too long.

pictorial-symbol-lost
A conversion essential to the message's delivery would cause one or more pictorial symbols to be lost.

punctuation-symbol-lost
A conversion essential to the message's delivery would cause one or more punctuation symbols to be lost.

reassignment-prohibited
The originator prohibited redirection to a recipient-assigned alternate recipient.

redirection-loop-detected
The message cannot be redirected to an alternate recipient because that recipient has previously redirected the message (that is, a redirection loop exists).

rendition-unsupported
A PDAU does not support a requested physical rendition attribute.

secure-messaging-error
The message cannot be transferred or delivered because that would violate the security policy in force.

submission-prohibited
Permission to submit messages to a DL is not granted to the originator, or to the DL of which the first DL is a member.

too-many-recipients
Too many recipients are specified.

5.4.6 Explicit Conversion

An instance of enumeration syntax `Explicit Conversion` identifies a type of conversion, defined by X.408, which a message's originator may request explicitly.

For 1984 its value is chosen from the following:
ia5-text-to-teletex, no-conversion, teletex-to-telex.

For 1988 its value may be chosen, additionally, from the following:
ia5-text-to-g3-fax, ia5-text-to-g4-class1, ia5-text-to-telex, ia5-text-to-videotex, teletex-to-g3-fax, teletex-to-g4-class1, teletex-to-ia5-text, teletex-to-videotex, telex-to-g3-fax, telex-to-g4-class1, telex-to-ia5-text, telex-to-teletex, videotex-to-ia5-text, videotex-to-teletex.

The value **no-conversion** signifies no conversion. Other values identify conversions by their names. Each piece of information of the type indicated before the word *to* is to be converted to information of the type indicated after the word *to*.

5.4.7 Postal Mode

An instance of enumeration syntax `Postal Mode` identifies a type of physical delivery. Its value is chosen from the following, the letters in parentheses denoting delivery modes defined in the referenced **Recommendation F.170**.

cc
Counter collection (D).

cc-with-telephone-advice
Counter collection with telephone advice (E).

cc-with-teletex-advice
Counter collection with Teletex advice (H).

cc-with-telex-advice
Counter collection with Telex advice (G).

express-mail
Express mail (C).

ordinary-mail
Ordinary mail (A).

special-delivery
Special delivery (B).

5.4.8 Postal Report

An instance of enumeration syntax `Postal Report` identifies a type of physical delivery report that may be issued for a recipient. Its value is chosen from the following:

notification-via-mts
Either physical delivery or physical delivery failure is reported via the MTS.

notification-via-mts-and-pds
Either physical delivery or physical delivery failure is reported via both the PDS and the MTS.

notification-via-pds
Either physical delivery or physical delivery failure is reported via the PDS.

undeliverable-mail-via-pds
Physical delivery failure (but not physical delivery) is reported via the PDS.

5.4.9 Priority

An instance of enumeration syntax `Priority` identifies a relative priority at which a message may be transferred. Its value is chosen from the following: **low**, **normal** and **urgent**.

5.4.10 Reason

An instance of enumeration syntax `Reason` explains in general terms why a message cannot be delivered to a recipient.

For 1984 its value is chosen from the following:

conversion-not-performed
A necessary conversion cannot be performed.

transfer-failed
A required communication cannot be effected.

transfer-impossible
The message itself is flawed.

For 1988 its value may be chosen, additionally, from the following:

directory-operation-failed
A required operation involving the directory system (DS) failed.

physical-delivery-not-done
A PDS cannot physically deliver the message.

physical-rendition-not-done
A PDAU cannot physically render the message.

restricted-delivery
The recipient has restricted delivery.

5.4.11 Redirection Reason

An instance of enumeration syntax `Redirection Reason` explains why a communique was redirected. Its value is chosen from the following:

originator-requested
The originator redirected the communique when either it could not be delivered to an intended recipient or the intended recipient was itself redirecting messages.

recipient-assigned
An intended recipient redirected the communique, and the originator had not prohibited such redirection.

recipient-directory-substitution
The communique could not be delivered to the intended recipient via the original O/R Address supplied. However, the O/R Name included a directory name and, from that directory, an alternate O/R Address for the recipient was obtained to which the communique was then delivered.

alias
The intended recipient's preferred address has not been specified. The message has been redirected to that preferred address.

recipient-domain-assigned
An intended recipient's MD redirected the communique when it found the recipient's O/R name malformed, and the originator had allowed such redirection.

5.4.12 Registration

An instance of enumeration syntax `Registration` identifies a kind of registered mail service that may be employed in a message's physical delivery. Its value is chosen from the following:

registered-mail
Registered mail with delivery to anyone at the recipient's address.

registered-mail-in-person
Registered mail with delivery only to the recipient.

unregistered-mail
Normal, unregistered mail.

5.4.13 Report Request

An instance of enumeration syntax `Report Request` identifies the circumstances under which a communique may provoke a report, as well as the nature of that report. Its value is chosen from the following:

always
A DR or an NDR is issued as appropriate. It comprises information from the final external trace entry alone.

always-audited
A DR or an NDR is issued as appropriate. It comprises all external trace entries.

never
Neither a DR nor an NDR is issued.

non-delivery
An NDR is issued, if appropriate, but a DR is not. The former comprises information from the final external trace entry alone.

5.4.14 Security Classification

An instance of enumeration syntax `Security Classification` classifies a communique or report for security purposes. Its value is chosen from the following, in order of increasing sensitivity: **unmarked**, **unclassified**, **restricted**, **confidential**, **secret** and **top-secret**. Also, the value may be **not_present** if no Security Classification is provided. The values used are dictated by the security policy in force.

5.4.15 Terminal Type

An instance of enumeration syntax `Terminal Type` identifies the type of a user's terminal. Its value is chosen from the following: **g3-fax**, **g4-fax**, **ia5-terminal**, **teletex**, **telex** and **videotex**. The meaning of each value is indicated by its name.

5.5 Declaration Summary

This section lists the declarations that define the portion of the C interface that deals with the MH packages.

The declarations, as assembled here, constitute the contents of a header file to be made accessible to client programmers. The header file is **<xmhp.h>**. The symbols the declarations define are the only MH symbols the service makes visible to the client.

Note: The identifier for the variable name of type OM_STRING of a class in the Message Handling package can usually be derived using the name of the class, preceded by "MH_C_", and replacing a blank space with an underscore. To be in line with the ANSI C language limitation, some words in the class names are excepted and are abbreviated as below:

BILATERAL_INFORMATION	is abbreviated to	BILATERAL_INFO,
DELIVERED		DELIV
CONFIRMATION		CONFIRM
CONFIRMATIONS		CONFIRMS
PER_RECIPIENT_		PER_RECIP_
DELIV_PER_RECIP_REPORT		DELIV_PER_RECIP_REP

```
/* BEGIN MH PORTION OF INTERFACE */

/* SYMBOLIC CONSTANTS */

/* Class */
#define OMP_O_MH_C_ALGORITHM              "\x2A\x86\x48\xCE\x22\x0B\x00"
#define OMP_O_MH_C_ALGORITHM_AND_RESULT   "\x2A\x86\x48\xCE\x22\x0B\x01"
#define OMP_O_MH_C_ASYMMETRIC_TOKEN       "\x2A\x86\x48\xCE\x22\x0B\x02"
#define OMP_O_MH_C_BILATERAL_INFO         "\x2A\x86\x48\xCE\x22\x0B\x03"
#define OMP_O_MH_C_COMMUNIQUE             "\x2A\x86\x48\xCE\x22\x0B\x04"
#define OMP_O_MH_C_CONTENT                "\x2A\x86\x48\xCE\x22\x0B\x05"
#define OMP_O_MH_C_DELIV_MESSAGE          "\x2A\x86\x48\xCE\x22\x0B\x06"
#define OMP_O_MH_C_DELIV_PER_RECIP_DR     "\x2A\x86\x48\xCE\x22\x0B\x07"
#define OMP_O_MH_C_DELIV_PER_RECIP_NDR    "\x2A\x86\x48\xCE\x22\x0B\x08"
#define OMP_O_MH_C_DELIV_PER_RECIP_REP    "\x2A\x86\x48\xCE\x22\x0B\x09"
#define OMP_O_MH_C_DELIV_REPORT           "\x2A\x86\x48\xCE\x22\x0B\x0A"
#define OMP_O_MH_C_DELIVERY_CONFIRM       "\x2A\x86\x48\xCE\x22\x0B\x0B"
#define OMP_O_MH_C_DELIVERY_ENVELOPE      "\x2A\x86\x48\xCE\x22\x0B\x0C"
#define OMP_O_MH_C_EITS                   "\x2A\x86\x48\xCE\x22\x0B\x0D"
#define OMP_O_MH_C_EXPANSION_RECORD       "\x2A\x86\x48\xCE\x22\x0B\x0E"
#define OMP_O_MH_C_EXTENSIBLE_OBJECT      "\x2A\x86\x48\xCE\x22\x0B\x0F"
#define OMP_O_MH_C_EXTENSION              "\x2A\x86\x48\xCE\x22\x0B\x10"
#define OMP_O_MH_C_EXTERNAL_TRACE_ENTRY   "\x2A\x86\x48\xCE\x22\x0B\x11"
#define OMP_O_MH_C_G3_FAX_NBPS            "\x2A\x86\x48\xCE\x22\x0B\x12"
#define OMP_O_MH_C_GENERAL_CONTENT        "\x2A\x86\x48\xCE\x22\x0B\x13"
#define OMP_O_MH_C_INTERNAL_TRACE_ENTRY   "\x2A\x86\x48\xCE\x22\x0B\x14"
#define OMP_O_MH_C_LOCAL_DELIV_CONFIRM    "\x2A\x86\x48\xCE\x22\x0B\x15"
#define OMP_O_MH_C_LOCAL_DELIV_CONFIRMS   "\x2A\x86\x48\xCE\x22\x0B\x16"
#define OMP_O_MH_C_LOCAL_NDR              "\x2A\x86\x48\xCE\x22\x0B\x17"
#define OMP_O_MH_C_LOCAL_PER_RECIP_NDR    "\x2A\x86\x48\xCE\x22\x0B\x18"
#define OMP_O_MH_C_MESSAGE                "\x2A\x86\x48\xCE\x22\x0B\x19"
#define OMP_O_MH_C_MESSAGE_RD             "\x2A\x86\x48\xCE\x22\x0B\x1A"
#define OMP_O_MH_C_MTS_IDENTIFIER         "\x2A\x86\x48\xCE\x22\x0B\x1B"
#define OMP_O_MH_C_OR_ADDRESS             "\x2A\x86\x48\xCE\x22\x0B\x1C"
#define OMP_O_MH_C_OR_NAME                "\x2A\x86\x48\xCE\x22\x0B\x1D"
```

```
#define OMP_O_MH_C_PER_RECIP_DR           "\x2A\x86\x48\xCE\x22\x0B\x22"
#define OMP_O_MH_C_PER_RECIP_NDR          "\x2A\x86\x48\xCE\x22\x0B\x1F"
#define OMP_O_MH_C_PER_RECIP_REPORT       "\x2A\x86\x48\xCE\x22\x0B\x20"
#define OMP_O_MH_C_PROBE                  "\x2A\x86\x48\xCE\x22\x0B\x21"
#define OMP_O_MH_C_PROBE_RD               "\x2A\x86\x48\xCE\x22\x0B\x22"
#define OMP_O_MH_C_RD                     "\x2A\x86\x48\xCE\x22\x0B\x23"
#define OMP_O_MH_C_REDIRECTION_RECORD     "\x2A\x86\x48\xCE\x22\x0B\x24"
#define OMP_O_MH_C_REPORT                 "\x2A\x86\x48\xCE\x22\x0B\x25"
#define OMP_O_MH_C_SECURITY_LABEL         "\x2A\x86\x48\xCE\x22\x0B\x26"
#define OMP_O_MH_C_SESSION                "\x2A\x86\x48\xCE\x22\x0B\x27"
#define OMP_O_MH_C_SUBMISSION_RESULTS     "\x2A\x86\x48\xCE\x22\x0B\x28"
#define OMP_O_MH_C_SUBMITTED_COMMUNIQUE   "\x2A\x86\x48\xCE\x22\x0B\x29"
#define OMP_O_MH_C_SUBMITTED_MESSAGE      "\x2A\x86\x48\xCE\x22\x0B\x2A"
#define OMP_O_MH_C_SUBMITTED_MESSAGE_RD   "\x2A\x86\x48\xCE\x22\x0B\x2B"
#define OMP_O_MH_C_SUBMITTED_PROBE        "\x2A\x86\x48\xCE\x22\x0B\x2C"
#define OMP_O_MH_C_SUBMITTED_PROBE_RD     "\x2A\x86\x48\xCE\x22\x0B\x2D"
#define OMP_O_MH_C_TELETEX_NBPS           "\x2A\x86\x48\xCE\x22\x0B\x2E"
#define OMP_O_MH_C_DELIVERY_REPORT        "\x2A\x86\x48\xCE\x22\x0B\x2F"
#define OMP_O_MH_C_MT_PUBLIC_DATA         "\x2A\x86\x48\xCE\x22\x0B\x30"
#define OMP_O_MH_C_TOKEN_PUBLIC_DATA      "\x2A\x86\x48\xCE\x22\x0B\x31"
#define OMP_O_MH_C_TOKEN                  "\x2A\x86\x48\xCE\x22\x0B\x32"

/* Enumeration */

/* Action */
#define MH_AC_EXPANDED          ( (OM_enumeration) -2 )
#define MH_AC_REDIRECTED        ( (OM_enumeration) -1 )
#define MH_AC_RELAYED           ( (OM_enumeration) 0 )
#define MH_AC_REROUTED          ( (OM_enumeration) 1 )
#define MH_AC_RECIP_REASSIGNED  ( (OM_enumeration) 2 )

/* Builtin EIT */
#define MH_BE_UNDEFINED         ( (OM_enumeration) 0 )
#define MH_BE_TELEX             ( (OM_enumeration) 1 )
#define MH_BE_IA5_TEXT          ( (OM_enumeration) 2 )
#define MH_BE_G3_FAX            ( (OM_enumeration) 3 )
#define MH_BE_G4_CLASS1         ( (OM_enumeration) 4 )
#define MH_BE_TELETEX           ( (OM_enumeration) 5 )
#define MH_BE_VIDEOTEX          ( (OM_enumeration) 6 )
#define MH_BE_MIXED_MODE        ( (OM_enumeration) 9 )
#define MH_BE_ODA               ( (OM_enumeration) 10 )
#define MH_BE_ISO_6937_TEXT     ( (OM_enumeration) 11 )

/* Delivery Mode */
#define MH_DM_ANY               ( (OM_enumeration) 0 )
#define MH_DM_MTS               ( (OM_enumeration) 1 )
#define MH_DM_PDS               ( (OM_enumeration) 2 )
#define MH_DM_TELEX             ( (OM_enumeration) 3 )
#define MH_DM_TELETEX           ( (OM_enumeration) 4 )
#define MH_DM_G3_FAX            ( (OM_enumeration) 5 )
#define MH_DM_G4_FAX            ( (OM_enumeration) 6 )
#define MH_DM_IA5_TERMINAL      ( (OM_enumeration) 7 )
#define MH_DM_VIDEOTEX          ( (OM_enumeration) 8 )
#define MH_DM_TELEPHONE         ( (OM_enumeration) 9 )

/* Delivery Point */
#define MH_DP_PUBLIC_UA         ( (OM_enumeration) 0 )
#define MH_DP_PRIVATE_UA        ( (OM_enumeration) 1 )
#define MH_DP_MS                ( (OM_enumeration) 2 )
```

```
#define MH_DP_DL                ( (OM_enumeration) 3 )
#define MH_DP_PDAU              ( (OM_enumeration) 4 )
#define MH_DP_PDS_PATRON        ( (OM_enumeration) 5 )
#define MH_DP_OTHER_AU          ( (OM_enumeration) 6 )

/* Diagnostic */
#define MH_DG_NO_DIAGNOSTIC               ( (OM_enumeration) -1 )
#define MH_DG_OR_NAME_UNRECOGNIZED        ( (OM_enumeration) 0 )
#define MH_DG_OR_NAME_AMBIGUOUS           ( (OM_enumeration) 1 )
#define MH_DG_MTS_CONGESTED               ( (OM_enumeration) 2 )
#define MH_DG_LOOP_DETECTED               ( (OM_enumeration) 3 )
#define MH_DG_RECIPIENT_UNAVAILABLE       ( (OM_enumeration) 4 )
#define MH_DG_MAXIMUM_TIME_EXPIRED        ( (OM_enumeration) 5 )
#define MH_DG_EITS_UNSUPPORTED            ( (OM_enumeration) 6 )
#define MH_DG_CONTENT_TOO_LONG            ( (OM_enumeration) 7 )
#define MH_DG_IMPRACTICAL_TO_CONVERT      ( (OM_enumeration) 8 )
#define MH_DG_PROHIBITED_TO_CONVERT       ( (OM_enumeration) 9 )
#define MH_DG_CONVERSION_UNSUBSCRIBED     ( (OM_enumeration) 10 )
#define MH_DG_PARAMETERS_INVALID          ( (OM_enumeration) 11 )
#define MH_DG_CONTENT_SYNTAX_IN_ERROR     ( (OM_enumeration) 12 )
#define MH_DG_LENGTH_CONSTRAINT_VIOLATD   ( (OM_enumeration) 13 )
#define MH_DG_NUMBER_CONSTRAINT_VIOLATD   ( (OM_enumeration) 14 )
#define MH_DG_CONTENT_TYPE_UNSUPPORTED    ( (OM_enumeration) 15 )
#define MH_DG_TOO_MANY_RECIPIENTS         ( (OM_enumeration) 16 )
#define MH_DG_NO_BILATERAL_AGREEMENT      ( (OM_enumeration) 17 )
#define MH_DG_CRITICAL_FUNC_UNSUPPORTED   ( (OM_enumeration) 18 )
#define MH_DG_CONVERSION_LOSS_PROHIB      ( (OM_enumeration) 19 )
#define MH_DG_LINE_TOO_LONG               ( (OM_enumeration) 20 )
#define MH_DG_PAGE_TOO_LONG               ( (OM_enumeration) 21 )
#define MH_DG_PICTORIAL_SYMBOL_LOST       ( (OM_enumeration) 22 )
#define MH_DG_PUNCTUATION_SYMBOL_LOST     ( (OM_enumeration) 23 )
#define MH_DG_ALPHABETIC_CHARACTER_LOST   ( (OM_enumeration) 24 )
#define MH_DG_MULTIPLE_INFO_LOSSES        ( (OM_enumeration) 25 )
#define MH_DG_REASSIGNMENT_PROHIBITED     ( (OM_enumeration) 26 )
#define MH_DG_REDIRECTION_LOOP_DETECTED   ( (OM_enumeration) 27 )
#define MH_DG_EXPANSION_PROHIBITED        ( (OM_enumeration) 28 )
#define MH_DG_SUBMISSION_PROHIBITED       ( (OM_enumeration) 29 )
#define MH_DG_EXPANSION_FAILED            ( (OM_enumeration) 30 )
#define MH_DG_RENDITION_UNSUPPORTED       ( (OM_enumeration) 31 )
#define MH_DG_MAIL_ADDRESS_INCORRECT      ( (OM_enumeration) 32 )
#define MH_DG_MAIL_OFFICE_INCOR_OR_INVD   ( (OM_enumeration) 33 )
#define MH_DG_MAIL_ADDRESS_INCOMPLETE     ( (OM_enumeration) 34 )
#define MH_DG_MAIL_RECIPIENT_UNKNOWN      ( (OM_enumeration) 35 )
#define MH_DG_MAIL_RECIPIENT_DECEASED     ( (OM_enumeration) 36 )
#define MH_DG_MAIL_ORGANIZATION_EXPIRED   ( (OM_enumeration) 37 )
#define MH_DG_MAIL_REFUSED                ( (OM_enumeration) 38 )
#define MH_DG_MAIL_UNCLAIMED              ( (OM_enumeration) 39 )
#define MH_DG_MAIL_RECIPIENT_MOVED        ( (OM_enumeration) 40 )
#define MH_DG_MAIL_RECIPIENT_TRAVELLING   ( (OM_enumeration) 41 )
#define MH_DG_MAIL_RECIPIENT_DEPARTED     ( (OM_enumeration) 42 )
#define MH_DG_MAIL_NEW_ADDRESS_UNKNOWN    ( (OM_enumeration) 43 )
#define MH_DG_MAIL_FORWARDING_UNWANTED    ( (OM_enumeration) 44 )
#define MH_DG_MAIL_FORWARDING_PROHIB      ( (OM_enumeration) 45 )
#define MH_DG_SECURE_MESSAGING_ERROR      ( (OM_enumeration) 46 )
#define MH_DG_DOWNGRADING_IMPOSSIBLE      ( (OM_enumeration) 47 )

/* Explicit Conversion */
#define MH_EC_NO_CONVERSION             ( (OM_enumeration) -1 )
#define MH_EC_IA5_TEXT_TO_TELETEX       ( (OM_enumeration) 0 )
```

```
#define MH_EC_TELETEX_TO_TELEX          ( (OM_enumeration) 1 )
#define MH_EC_TELEX_TO_IA5_TEXT         ( (OM_enumeration) 2 )
#define MH_EC_TELEX_TO_TELETEX          ( (OM_enumeration) 3 )
#define MH_EC_TELEX_TO_G4_CLASS1        ( (OM_enumeration) 4 )
#define MH_EC_IA5_TEXT_TO_TELEX         ( (OM_enumeration) 6 )
#define MH_EC_TELEX_TO_G3_FAX           ( (OM_enumeration) 7 )
#define MH_EC_IA5_TEXT_TO_G3_FAX        ( (OM_enumeration) 8 )
#define MH_EC_IA5_TEXT_TO_G4_CLASS1     ( (OM_enumeration) 9 )
#define MH_EC_IA5_TEXT_TO_VIDEOTEX      ( (OM_enumeration) 10 )
#define MH_EC_TELETEX_TO_IA5_TEXT       ( (OM_enumeration) 11 )
#define MH_EC_TELETEX_TO_G3_FAX         ( (OM_enumeration) 12 )
#define MH_EC_TELETEX_TO_G4_CLASS1      ( (OM_enumeration) 13 )
#define MH_EC_TELETEX_TO_VIDEOTEX       ( (OM_enumeration) 14 )
#define MH_EC_VIDEOTEX_TO_IA5_TEXT      ( (OM_enumeration) 16 )
#define MH_EC_VIDEOTEX_TO_TELETEX       ( (OM_enumeration) 17 )

/* Postal Mode */
#define MH_PM_ORDINARY_MAIL             ( (OM_enumeration) 0 )
#define MH_PM_SPECIAL_DELIVERY          ( (OM_enumeration) 1 )
#define MH_PM_EXPRESS_MAIL              ( (OM_enumeration) 2 )
#define MH_PM_CC                        ( (OM_enumeration) 3 )
#define MH_PM_CC_WITH_TELEPHONE_ADVICE  ( (OM_enumeration) 4 )
#define MH_PM_CC_WITH_TELEX_ADVICE      ( (OM_enumeration) 5 )
#define MH_PM_CC_WITH_TELETEX_ADVICE    ( (OM_enumeration) 6 )

/* Postal Report */
#define MH_PR_UNDELIVBLE_MAIL_VIA_PDS   ( (OM_enumeration) 0 )
#define MH_PR_NOTIFICN_VIA_PDS          ( (OM_enumeration) 1 )
#define MH_PR_NOTIFICN_VIA_MTS          ( (OM_enumeration) 2 )
#define MH_PR_NOTIFICN_VIA_MTS_AND_PDS  ( (OM_enumeration) 3 )

/* Priority */
#define MH_PTY_NORMAL                   ( (OM_enumeration) 0 )
#define MH_PTY_LOW                      ( (OM_enumeration) 1 )
#define MH_PTY_URGENT                   ( (OM_enumeration) 2 )

/* Reason */
#define MH_RE_TRANSFER_FAILED            ( (OM_enumeration) 0 )
#define MH_RE_TRANSFER_IMPOSSIBLE        ( (OM_enumeration) 1 )
#define MH_RE_CONVERSION_NOT_PERFORMED   ( (OM_enumeration) 2 )
#define MH_RE_PHYSICAL_RENDITN_NOT_DONE  ( (OM_enumeration) 3 )
#define MH_RE_PHYSICAL_DELIV_NOT_DONE    ( (OM_enumeration) 4 )
#define MH_RE_RESTRICTED_DELIVERY        ( (OM_enumeration) 5 )
#define MH_RE_DIRECTORY_OPERATN_FAILED   ( (OM_enumeration) 6 )

/* Redirection Reason */
#define MH_RR_RECIPIENT_ASSIGNED         ( (OM_enumeration) 0 )
#define MH_RR_ORIGINATOR_REQUESTED       ( (OM_enumeration) 1 )
#define MH_RR_RECIPIENT_DOMAIN_ASSIGNED  ( (OM_enumeration) 2 )
#define MH_RR_RECIPIENT_DIRECTORY_SUBS   ( (OM_enumeration) 3 )
#define MH_RR_ALIAS                      ( (OM_enumeration) 4 )

/* Registration */
#define MH_RG_UNREGISTERED_MAIL          ( (OM_enumeration) 0 )
#define MH_RG_REGISTERED_MAIL            ( (OM_enumeration) 1 )
#define MH_RG_REGISTERED_MAIL_IN_PERSON  ( (OM_enumeration) 2 )

/* Report Request */
#define MH_RQ_NEVER                      ( (OM_enumeration) 0 )
```

```
#define MH_RQ_NON_DELIVERY              ( (OM_enumeration) 1 )
#define MH_RQ_ALWAYS                    ( (OM_enumeration) 2 )
#define MH_RQ_ALWAYS_AUDITED            ( (OM_enumeration) 3 )

/* Security Classification */
#define MH_SC_NOT_PRESENT               ( (OM_enumeration) -1 )
#define MH_SC_UNMARKED                  ( (OM_enumeration) 0 )
#define MH_SC_UNCLASSIFIED              ( (OM_enumeration) 1 )
#define MH_SC_RESTRICTED                ( (OM_enumeration) 2 )
#define MH_SC_CONFIDENTIAL              ( (OM_enumeration) 3 )
#define MH_SC_SECRET                    ( (OM_enumeration) 4 )
#define MH_SC_TOP_SECRET                ( (OM_enumeration) 5 )

/* Terminal Type */
#define MH_TT_TELEX                     ( (OM_enumeration) 3 )
#define MH_TT_TELETEX                   ( (OM_enumeration) 4 )
#define MH_TT_G3_FAX                    ( (OM_enumeration) 5 )
#define MH_TT_G4_FAX                    ( (OM_enumeration) 6 )
#define MH_TT_IA5_TERMINAL              ( (OM_enumeration) 7 )
#define MH_TT_VIDEOTEX                  ( (OM_enumeration) 8 )

/* Integer */

/* Content Type */
#define MH_CTI_UNIDENTIFIED                 ( (OM_integer) 0 )
#define MH_CTI_EXTERNAL                     ( (OM_integer) 1 )
#define MH_CTI_P2_1984                      ( (OM_integer) 2 )
#define MH_CTI_P2_1988                      ( (OM_integer) 22 )


/* Object Identifier (Elements component) */

/* Content Type */
#define OMP_O_MH_CTO_INNER_MESSAGE          "\x56\x03\x03\x01"
#define OMP_O_MH_CTO_UNIDENTIFIED           "\x56\x03\x03\x00"

/* External EITs */
#define OMP_O_MH_EE_G3_FAX                  "\x56\x03\x04\x03"
#define OMP_O_MH_EE_G4_CLASS_1              "\x56\x03\x04\x04"
#define OMP_O_MH_EE_IA5_TEXT                "\x56\x03\x04\x02"
#define OMP_O_MH_EE_MIXED_MODE              "\x56\x03\x04\x09"
#define OMP_O_MH_EE_TELETEX                 "\x56\x03\x04\x05"
#define OMP_O_MH_EE_TELEX                   "\x56\x03\x04\x01"
#define OMP_O_MH_EE_UNDEFINED               "\x56\x03\x04\x00"
#define OMP_O_MH_EE_VIDEOTEX                "\x56\x03\x04\x06"

/* ODA EIT */
#define OMP_O_MH_ET_ODA_DATA                "\x58\x01\x00\x01"

/* Rendition Attributes */
#define OMP_O_MH_RA_BASIC_RENDITION         "\x56\x03\x05\x00"

/* Type */

#define MH_T_A3_WIDTH                   ( (OM_type) 200 )
#define MH_T_ACTION                     ( (OM_type) 201 )
#define MH_T_ACTUAL_RECIPIENT_NAME      ( (OM_type) 202 )
#define MH_T_ADMD_NAME                  ( (OM_type) 203 )
#define MH_T_ALGORITHM_DATUM            ( (OM_type) 204 )
```

```
#define MH_T_ALGORITHM_ID                ( (OM_type) 205 )
#define MH_T_ALGORITHM_RESULT            ( (OM_type) 206 )
#define MH_T_ALTERNATE_RECIP_ALLOWED     ( (OM_type) 207 )
#define MH_T_ALTERNATE_RECIPIENT_NAME    ( (OM_type) 208 )
#define MH_T_ARRIVAL_TIME                ( (OM_type) 209 )
#define MH_T_ATTEMPTED_ADMD_NAME         ( (OM_type) 210 )
#define MH_T_ATTEMPTED_COUNTRY_NAME      ( (OM_type) 211 )
#define MH_T_ATTEMPTED_MTA_NAME          ( (OM_type) 212 )
#define MH_T_ATTEMPTED_PRMD_IDENTIFIER   ( (OM_type) 213 )
#define MH_T_B4_LENGTH                   ( (OM_type) 214 )
#define MH_T_B4_WIDTH                    ( (OM_type) 215 )
#define MH_T_BILATERAL_INFO              ( (OM_type) 216 )
#define MH_T_BINARY_CONTENT              ( (OM_type) 217 )
#define MH_T_BUILTIN_EITS                ( (OM_type) 218 )
#define MH_T_BUREAU_FAX_DELIVERY         ( (OM_type) 219 )
#define MH_T_COMMON_NAME                 ( (OM_type) 220 )
#define MH_T_CONFIDENTIALITY_ALGORITHM   ( (OM_type) 221 )
#define MH_T_CONTENT                     ( (OM_type) 223 )
#define MH_T_CONTENT_CORRELATOR          ( (OM_type) 224 )
#define MH_T_CONTENT_EXTENSIONS          ( (OM_type) 225 )
#define MH_T_CONTENT_IDENTIFIER          ( (OM_type) 226 )
#define MH_T_CONTENT_LENGTH              ( (OM_type) 227 )
#define MH_T_CONTENT_RETURN_REQUESTED    ( (OM_type) 228 )
#define MH_T_CONTENT_TYPE                ( (OM_type) 229 )
#define MH_T_CONTROL_CHARACTER_SETS      ( (OM_type) 230 )
#define MH_T_CONVERSION_LOSS_PROHIBITED  ( (OM_type) 231 )
#define MH_T_CONVERSION_PROHIBITED       ( (OM_type) 232 )
#define MH_T_CONVERTED_EITS              ( (OM_type) 233 )
#define MH_T_COUNTRY_NAME                ( (OM_type) 234 )
#define MH_T_CRITICAL_FOR_DELIVERY       ( (OM_type) 235 )
#define MH_T_CRITICAL_FOR_SUBMISSION     ( (OM_type) 236 )
#define MH_T_CRITICAL_FOR_TRANSFER       ( (OM_type) 237 )
#define MH_T_DEFERRED_DELIVERY_TIME      ( (OM_type) 238 )
#define MH_T_DEFERRED_TIME               ( (OM_type) 239 )
#define MH_T_DELIVERY_CONFIRMS           ( (OM_type) 240 )
#define MH_T_DELIVERY_POINT              ( (OM_type) 241 )
#define MH_T_DELIVERY_TIME               ( (OM_type) 242 )
#define MH_T_DIRECTORY_NAME              ( (OM_type) 243 )
#define MH_T_DISCLOSURE_ALLOWED          ( (OM_type) 244 )
#define MH_T_DISTINGUISHED_RECIP_ADDR    ( (OM_type) 245 )
#define MH_T_DOMAIN_TYPE_1               ( (OM_type) 246 )
#define MH_T_DOMAIN_TYPE_2               ( (OM_type) 247 )
#define MH_T_DOMAIN_TYPE_3               ( (OM_type) 248 )
#define MH_T_DOMAIN_TYPE_4               ( (OM_type) 249 )
#define MH_T_DOMAIN_VALUE_1              ( (OM_type) 250 )
#define MH_T_DOMAIN_VALUE_2              ( (OM_type) 251 )
#define MH_T_DOMAIN_VALUE_3              ( (OM_type) 252 )
#define MH_T_DOMAIN_VALUE_4              ( (OM_type) 253 )
#define MH_T_ENVELOPES                   ( (OM_type) 254 )
#define MH_T_EVENT_HANDLE                ( (OM_type) 255 )
#define MH_T_EXPANSION_HISTORY           ( (OM_type) 256 )
#define MH_T_EXPANSION_PROHIBITED        ( (OM_type) 257 )
#define MH_T_EXPLICIT_CONVERSION         ( (OM_type) 258 )
#define MH_T_EXTENSION_TYPE              ( (OM_type) 259 )
#define MH_T_EXTENSION_VALUE             ( (OM_type) 260 )
#define MH_T_EXTENSIONS                  ( (OM_type) 261 )
#define MH_T_EXTERNAL_EITS               ( (OM_type) 262 )
#define MH_T_EXTERNAL_TRACE_INFO         ( (OM_type) 263 )
#define MH_T_FINE_RESOLUTION             ( (OM_type) 264 )
```

```
#define MH_T_FORWARDING_ADDRESS          ( (OM_type) 265 )
#define MH_T_FORWARDING_ADDR_REQUESTED   ( (OM_type) 266 )
#define MH_T_FORWARDING_PROHIBITED       ( (OM_type) 267 )
#define MH_T_G3_FAX_NBPS                 ( (OM_type) 268 )
#define MH_T_G4_FAX_NBPS                 ( (OM_type) 269 )
#define MH_T_GENERATION                  ( (OM_type) 270 )
#define MH_T_GIVEN_NAME                  ( (OM_type) 271 )
#define MH_T_GRAPHIC_CHARACTER_SETS      ( (OM_type) 272 )
#define MH_T_INFORMATION                 ( (OM_type) 273 )
#define MH_T_INITIALS                    ( (OM_type) 274 )
#define MH_T_INTEGRITY_CHECK             ( (OM_type) 275 )
#define MH_T_INTENDED_RECIPIENT_NAME     ( (OM_type) 276 )
#define MH_T_INTENDED_RECIPIENT_NUMBER   ( (OM_type) 277 )
#define MH_T_INTERNAL_TRACE_INFO         ( (OM_type) 278 )
#define MH_T_ISDN_NUMBER                 ( (OM_type) 279 )
#define MH_T_ISDN_SUBADDRESS             ( (OM_type) 280 )
#define MH_T_LATEST_DELIVERY_TIME        ( (OM_type) 281 )
#define MH_T_LOCAL_IDENTIFIER            ( (OM_type) 282 )
#define MH_T_MESSAGE_SEQUENCE_NUMBER     ( (OM_type) 283 )
#define MH_T_MISCELANEOUS_CAPABILITIES   ( (OM_type) 284 )
#define MH_T_MTA_CERTIFICATE             ( (OM_type) 285 )
#define MH_T_MTA_NAME                    ( (OM_type) 286 )
#define MH_T_MTA_REPORT_REQUEST          ( (OM_type) 287 )
#define MH_T_MTA_RESPONSIBILITY          ( (OM_type) 288 )
#define MH_T_MTS_IDENTIFIER              ( (OM_type) 289 )
#define MH_T_NAME                        ( (OM_type) 290 )
#define MH_T_NON_DELIVERY_DIAGNOSTIC     ( (OM_type) 291 )
#define MH_T_NON_DELIVERY_REASON         ( (OM_type) 292 )
#define MH_T_NUMERIC_USER_IDENTIFIER     ( (OM_type) 293 )
#define MH_T_ORGANIZATION_NAME           ( (OM_type) 294 )
#define MH_T_ORGANIZATIONAL_UNIT_NAME_1  ( (OM_type) 295 )
#define MH_T_ORGANIZATIONAL_UNIT_NAME_2  ( (OM_type) 296 )
#define MH_T_ORGANIZATIONAL_UNIT_NAME_3  ( (OM_type) 297 )
#define MH_T_ORGANIZATIONAL_UNIT_NAME_4  ( (OM_type) 298 )
#define MH_T_ORIG_AND_EXPANSION_HISTORY  ( (OM_type) 299 )
#define MH_T_ORIGIN_CHECK                ( (OM_type) 300 )
#define MH_T_ORIGINAL_EITS               ( (OM_type) 301 )
#define MH_T_ORIGINALLY_INTENDED_RECIP   ( (OM_type) 302 )
#define MH_T_ORIGINATOR_CERTIFICATE      ( (OM_type) 303 )
#define MH_T_ORIGINATOR_NAME             ( (OM_type) 304 )
#define MH_T_ORIGINATOR_REPORT_REQUEST   ( (OM_type) 305 )
#define MH_T_ORIGINATOR_RETURN_ADDRESS   ( (OM_type) 306 )
#define MH_T_OTHER_RECIPIENT_NAMES       ( (OM_type) 307 )
#define MH_T_PAGE_FORMATS                ( (OM_type) 308 )
#define MH_T_PER_RECIP_REPORTS           ( (OM_type) 309 )
#define MH_T_POSTAL_ADDRESS_DETAILS      ( (OM_type) 310 )
#define MH_T_POSTAL_ADDRESS_IN_FULL      ( (OM_type) 311 )
#define MH_T_POSTAL_ADDRESS_IN_LINES     ( (OM_type) 312 )
#define MH_T_POSTAL_CODE                 ( (OM_type) 313 )
#define MH_T_POSTAL_COUNTRY_NAME         ( (OM_type) 314 )
#define MH_T_POSTAL_DELIVERY_POINT_NAME  ( (OM_type) 315 )
#define MH_T_POSTAL_DELIV_SYSTEM_NAME    ( (OM_type) 316 )
#define MH_T_POSTAL_GENERAL_DELIV_ADDR   ( (OM_type) 317 )
#define MH_T_POSTAL_LOCALE               ( (OM_type) 318 )
#define MH_T_POSTAL_MODE                 ( (OM_type) 319 )
#define MH_T_POSTAL_OFFICE_BOX_NUMBER    ( (OM_type) 320 )
#define MH_T_POSTAL_OFFICE_NAME          ( (OM_type) 321 )
#define MH_T_POSTAL_OFFICE_NUMBER        ( (OM_type) 322 )
#define MH_T_POSTAL_ORGANIZATION_NAME    ( (OM_type) 323 )
```

```
#define MH_T_POSTAL_PATRON_DETAILS        ( (OM_type) 324 )
#define MH_T_POSTAL_PATRON_NAME           ( (OM_type) 325 )
#define MH_T_POSTAL_REPORT                ( (OM_type) 326 )
#define MH_T_POSTAL_STREET_ADDRESS        ( (OM_type) 327 )
#define MH_T_PREFERRED_DELIVERY_MODES     ( (OM_type) 328 )
#define MH_T_PRESENTATION_ADDRESS         ( (OM_type) 329 )
#define MH_T_PRIORITY                     ( (OM_type) 330 )
#define MH_T_PRIVACY_MARK                 ( (OM_type) 331 )
#define MH_T_PRIVATE_USE                  ( (OM_type) 332 )
#define MH_T_PRMD_IDENTIFIER              ( (OM_type) 333 )
#define MH_T_PRMD_NAME                    ( (OM_type) 334 )
#define MH_T_PROOF_OF_DELIVERY            ( (OM_type) 335 )
#define MH_T_PROOF_OF_DELIV_REQUESTED     ( (OM_type) 336 )
#define MH_T_PROOF_OF_SUBMISSION          ( (OM_type) 337 )
#define MH_T_PROOF_OF_SUBMISN_REQUEST     ( (OM_type) 338 )
#define MH_T_PUBLIC_INFORMATION           ( (OM_type) 339 )
#define MH_T_REASON                       ( (OM_type) 341 )
#define MH_T_REASSIGNMENT_PROHIBITED      ( (OM_type) 342 )
#define MH_T_RECIPIENT_CERTIFICATE        ( (OM_type) 343 )
#define MH_T_RECIPIENT_DESCRIPTORS        ( (OM_type) 344 )
#define MH_T_RECIPIENT_NAME               ( (OM_type) 345 )
#define MH_T_RECIPIENT_NUMBER             ( (OM_type) 346 )
#define MH_T_RECIP_NUMBER_FOR_ADVICE      ( (OM_type) 347 )
#define MH_T_REDIRECTION_HISTORY          ( (OM_type) 348 )
#define MH_T_REGISTRATION                 ( (OM_type) 349 )
#define MH_T_RENDITION_ATTRIBUTES         ( (OM_type) 350 )
#define MH_T_REPORT_ADDITIONAL_INFO       ( (OM_type) 351 )
#define MH_T_REPORT_DESTINATION           ( (OM_type) 352 )
#define MH_T_REPORTING_DL_NAME            ( (OM_type) 353 )
#define MH_T_REPORTING_MTA_CERTIFICATE    ( (OM_type) 354 )
#define MH_T_SECRET_INFORMATION           ( (OM_type) 355 )
#define MH_T_SECURITY_CATEGORY_DATA       ( (OM_type) 356 )
#define MH_T_SECURITY_CATEGORY_IDS        ( (OM_type) 357 )
#define MH_T_SECURITY_CLASSIFICATION      ( (OM_type) 358 )
#define MH_T_SECURITY_LABEL               ( (OM_type) 359 )
#define MH_T_SECURITY_POLICY_ID           ( (OM_type) 360 )
#define MH_T_SIGNATURE                    ( (OM_type) 361 )
#define MH_T_SUBJECT_EXT_TRACE_INFO       ( (OM_type) 362 )
#define MH_T_SUBJECT_MTS_IDENTIFIER       ( (OM_type) 363 )
#define MH_T_SUBMISSION_TIME              ( (OM_type) 364 )
#define MH_T_SUPPLEMENTARY_INFO           ( (OM_type) 365 )
#define MH_T_SURNAME                      ( (OM_type) 366 )
#define MH_T_TELETEX_NBPS                 ( (OM_type) 367 )
#define MH_T_TEMPORARY                    ( (OM_type) 368 )
#define MH_T_TERMINAL_IDENTIFIER          ( (OM_type) 369 )
#define MH_T_TERMINAL_TYPE                ( (OM_type) 370 )
#define MH_T_TIME                         ( (OM_type) 371 )
#define MH_T_TOKEN                        ( (OM_type) 372 )
#define MH_T_TWO_DIMENSIONAL              ( (OM_type) 373 )
#define MH_T_UNCOMPRESSED                 ( (OM_type) 374 )
#define MH_T_UNLIMITED_LENGTH             ( (OM_type) 375 )
#define MH_T_X121_ADDRESS                 ( (OM_type) 377 )

/* Value Length */

#define MH_VL_ADMD_NAME                   ( (OM_value_length) 16 )
#define MH_VL_ATTEMPTED_ADMD_NAME         ( (OM_value_length) 16 )
#define MH_VL_ATTEMPTED_COUNTRY_NAME      ( (OM_value_length) 3 )
#define MH_VL_ATTEMPTED_MTA_NAME          ( (OM_value_length) 32 )
```

```
#define MH_VL_ATTEMPTED_PRMD_IDENTIFIER   ( (OM_value_length) 16 )
#define MH_VL_COMMON_NAME                 ( (OM_value_length) 64 )
#define MH_VL_CONTENT_CORRELATOR          ( (OM_value_length) 512 )
#define MH_VL_CONTENT_IDENTIFIER          ( (OM_value_length) 16 )
#define MH_VL_COUNTRY_NAME                ( (OM_value_length) 3 )
#define MH_VL_DOMAIN_TYPE                 ( (OM_value_length) 8 )
#define MH_VL_DOMAIN_VALUE                ( (OM_value_length) 128 )
#define MH_VL_GENERATION                  ( (OM_value_length) 3 )
#define MH_VL_GIVEN_NAME                  ( (OM_value_length) 16 )
#define MH_VL_INFORMATION                 ( (OM_value_length) 1024 )
#define MH_VL_INITIALS                    ( (OM_value_length) 5 )
#define MH_VL_ISDN_NUMBER                 ( (OM_value_length) 15 )
#define MH_VL_ISDN_SUBADDRESS             ( (OM_value_length) 40 )
#define MH_VL_LATEST_DELIVERY_TIME        ( (OM_value_length) 7 )
#define MH_VL_LOCAL_IDENTIFIER            ( (OM_value_length) 32 )
#define MH_VL_MTA_NAME                    ( (OM_value_length) 32 )
#define MH_VL_NUMERIC_USER_IDENTIFIER     ( (OM_value_length) 32 )
#define MH_VL_ORGANIZATION_NAME           ( (OM_value_length) 64 )
#define MH_VL_ORGANIZATIONAL_UNIT_NAMES   ( (OM_value_length) 32 )
#define MH_VL_POSTAL_ADDRESS_DETAILS      ( (OM_value_length) 30 )
#define MH_VL_POSTAL_ADDRESS_IN_FULL      ( (OM_value_length) 185 )
#define MH_VL_POSTAL_CODE                 ( (OM_value_length) 16 )
#define MH_VL_POSTAL_COUNTRY_NAME         ( (OM_value_length) 32 )
#define MH_VL_POSTAL_DELIV_POINT_NAME     ( (OM_value_length) 30 )
#define MH_VL_POSTAL_DELIV_SYSTEM_NAME    ( (OM_value_length) 16 )
#define MH_VL_POSTAL_GENERAL_DELIV_ADDR   ( (OM_value_length) 30 )
#define MH_VL_POSTAL_LOCALE               ( (OM_value_length) 30 )
#define MH_VL_POSTAL_OFFICE_BOX_NUMBER    ( (OM_value_length) 30 )
#define MH_VL_POSTAL_OFFICE_NAME          ( (OM_value_length) 30 )
#define MH_VL_POSTAL_OFFICE_NUMBER        ( (OM_value_length) 30 )
#define MH_VL_POSTAL_ORGANIZATION_NAME    ( (OM_value_length) 30 )
#define MH_VL_POSTAL_PATRON_DETAILS       ( (OM_value_length) 30 )
#define MH_VL_POSTAL_PATRON_NAME          ( (OM_value_length) 30 )
#define MH_VL_POSTAL_STREET_ADDRESS       ( (OM_value_length) 30 )
#define MH_VL_PRIVACY_MARK                ( (OM_value_length) 128 )
#define MH_VL_PRIVATE_USE                 ( (OM_value_length) 126 )
#define MH_VL_PRMD_IDENTIFIER             ( (OM_value_length) 16 )
#define MH_VL_PRMD_NAME                   ( (OM_value_length) 16 )
#define MH_VL_RECIP_NUMBER_FOR_ADVICE     ( (OM_value_length) 32 )
#define MH_VL_REDIRECTION_TIME            ( (OM_value_length) 17 )
#define MH_VL_REPORT_ADDITIONAL_INFO      ( (OM_value_length) 1024 )
#define MH_VL_SUPPLEMENTARY_INFO          ( (OM_value_length) 64 )
#define MH_VL_SURNAME                     ( (OM_value_length) 40 )
#define MH_VL_TERMINAL_IDENTIFIER         ( (OM_value_length) 24 )
#define MH_VL_TIME                        ( (OM_value_length) 17 )
#define MH_VL_X121_ADDRESS                ( (OM_value_length) 15 )

/* Value Number */

#define MH_VN_BILATERAL_INFORMATION       ( (OM_uint32) 512 )
#define MH_VN_BILATERAL_INFORMATION_84    ( (OM_uint32) 8 )
#define MH_VN_ENCODED_INFORMATION_TYPES   ( (OM_uint32) 10 )
#define MH_VN_EXPANSION_HISTORY           ( (OM_uint32) 512 )
#define MH_VN_OTHER_RECIPIENT_NAMES       ( (OM_uint32) 32767 )
#define MH_VN_PREFERRED_DELIVERY_MODES    ( (OM_uint32) 10 )
#define MH_VN_RECIPIENT_DESCRIPTORS       ( (OM_uint32) 32767 )
#define MH_VN_REDIRECTION_HISTORY         ( (OM_uint32) 512 )
#define MH_VN_PER_RECIP_REPORTS           ( (OM_uint32) 32767 )
#define MH_VN_SECURITY_CATEGORY_DATA      ( (OM_uint32) 64 )
```

```
#define MH_VN_SECURITY_CATEGORY_IDS       ( (OM_uint32) 64 )
#define MH_VN_TRACE_INFO                  ( (OM_uint32) 512 )

/* END MH PORTION OF INTERFACE */
```

Chapter 6

Secure Messaging Package

6.1 Summary

This chapter defines the SM 88 Package, which provides the functionality of SM (1988). The chapter is to be understood in the context provided by the referenced **XOM** Specification.

Instances of the SM classes are not conveyed between the client and the service by means of the interface, but rather serve as arguments of mathematical (typically cryptographic) algorithms.

Throughout this chapter, the words *originator* and *recipient* refer to the roles that various users, by means of their UAs, play in the conveyance of communiques and reports via the MTS. Furthermore, the term *subject message* used in reference to a communique denotes either that communique, if it is a message, or any of the messages denoted by the communique, if it is a probe.

6.2 Class Hierarchy

This section depicts the hierarchical organisation of the SM classes. Subclassification is indicated by indentation. The names of abstract classes are in italics. Thus, for example, Integrity Check Basis is an immediate subclass of Object, an abstract class. The names of classes to which the *om_encode*( ) function applies are in bold. The *om_create*( ) function applies to all concrete classes.

Object (defined in the referenced **XOM** Specification)
- **Integrity Check Basis**
- **Origin Check Basis**
- *Per-recipient Check Basis*
  - — **Per-recipient Delivery Check Basis**
  - — **Per-recipient Non-delivery Check Basis**
- **Proof of Delivery Basis**
- **Proof of Submission Basis**
- *Token Secret Data*
  - — **MT Secret Data**

Note: The identifier for the variable name of type OM_STRING of a class in the Secure Messaging package can usually be derived using the name of the class, preceded by ‘‘SM_C_’’, and replacing a blank space with an underscore. To be in line with the ANSI C language limitation, some words in the class names are excepted and are abbreviated as below:

RECIPIENT	is abbreviated as	RECIP
NON_DEL		NO_DEL
BASIS		is ignored

6.3 Class Definitions

This section defines the SM classes. It describes the attributes specific to a class in a table like those used in the referenced **XOM** Specification. The table includes, however, an additional column that identifies the attributes that are for 1988 alone.

Note: The additional column is included in the tables in this chapter for documentation consistency. In fact, all of the attributes specific to all of the SM classes are for 1988 alone.

6.3.1 Integrity Check Basis

An instance of class **Integrity Check Basis** provides the basis for enabling any recipient of a message to verify that its content is unchanged.

The algorithm used to compute the integrity check may be either symmetric or asymmetric. If the former, the check is computed by the originator and validated by the recipient using a single key which can be derived from the message's Token attribute or distributed by other means. If the latter, the check is computed by the originator using its secret key, and validated by the recipient using the associated public key which can be derived from the message's Originator Certificate attribute.

The attributes specific to this class are listed in the following table.

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
Algorithm	Object (Algorithm)	-	1	-	1988
Content	String (Octet)	-	1	-	1988

Table 6-1 Attributes Specific to Integrity Check Basis

Algorithm
Identifies the algorithm used to compute the integrity check.

Content
The unencrypted content whose integrity is being checked.

6.3.2 Origin Check Basis

An instance of class **Origin Check Basis** provides the basis for enabling a third party (for example, a user or an MTA) to verify the origin of portions of a communique or a report. For a message, the origin check also constitutes an integrity check and provides proof of the association between the content and the message's Security Label attribute. If it is based on the unencrypted version of the content (see below), the check also provides for non-repudiation of the content's origin. For a probe, the check also provides proof of the association between the probe's Content Identifier and Security Label attributes. For a report, the check also provides for verification that the report is unchanged, proof of the association between the report and its Security Label attribute, and non-repudiation of the report's origin.

The algorithm used to compute the check is asymmetric. The check is computed by its originator using its secret key, and validated by the third party using the associated public key which can be derived from the communique's Originator Certificate attribute or the report's Reporting MTA Certificate attribute.

The attributes specific to this class are listed in the following table.

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
Algorithm	Object (Algorithm)	-	1	-	1988
Content	String (Octet)	-	0-1	-	1988
Content Identifier	String (Printable)	1-16	0-1	-	1988
Per-recipient Checks	Object (Per-recepient Check Basis)	-	0-32767	-	1988
Security Label	Object (Security Label)	-	0-1	-	1988

Table 6-2 Attributes Specific to Origin Check Basis

Algorithm
Identifies the algorithm used to compute the origin check.

Content
The encrypted or unencrypted version of the like-named attribute of the message whose origin is being established: unencrypted if confidentiality is being provided (to allow anyone to validate the origin check), encrypted otherwise. This attribute shall be present if, and only if, the origin of a message is being verified.

Content Identifier
The like-named attribute, if any, of the communique whose origin is being verified or of the subject message of the report whose origin is being verified.

Per-recipient Checks
Establishes the origin of certain information about the recipients to which the origin check applies. The algorithm involved is applied to an instance of the Per-recipient Check Basis class. This attribute shall be present if, and only if, the origin of a report is being established.

Security Label
The like-named attribute, if any, of the communique or report whose origin is being verified.

6.3.3 Per-recipient Check Basis

An instance of class *Per-recipient Check Basis* provides the basis for enabling a third party (for example, a user or an MTA) to verify the origin of a portion of a per-recipient report.

The attributes specific to this class are listed in the following table.

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
Actual Recipient Name	Object (OR Name)	-	1	-	1988
Originally Intended Recip	Object (OR Name)	-	0-1	-	1988

Table 6-3 Attributes Specific to Per-recipient Check Basis

Actual Recipient Name
The O/R name of the recipient to which the per-recipient report applies. If the per-recipient report concerns a message (not a probe) and the recipient is an alternate recipient, this attribute's value is the O/R name of that alternate recipient.

Originally Intended Recip
The O/R name of the originally intended recipient. This attribute is present if, and only if, the Actual Recipient Name attribute denotes an alternate recipient.

6.3.4 Per-recipient Delivery Check Basis

An instance of class **Per-recipient Delivery Check Basis** provides the basis for enabling a third party (for example, a user or an MTA) to verify the origin of a portion of a per-recipient DR.

The attributes specific to this class are listed in the following table.

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
Delivery Point	Enum (Delivery Point)	-	1	private-UA	1988
Delivery Time	String (UTC Time)	0-17	1	-	1988
Proof of Delivery	Object (Algorithm and Result)	-	0-1	-	1988
Recipient Certificate	Object (Certificates)[1]	-	0-1	-	1988

[1] As defined in the referenced **XDS** Specification.

Table 6-4 Attributes Specific to Per-recipient Delivery Check Basis

Delivery Point
The nature of the functional entity by means of which the subject message was, or would have been, delivered to the recipient. For its defined values, see Section 5.4.4 on page 144.

Delivery Time
The date and time at which the subject message was or would have been delivered to the recipient.

Proof of Delivery
Proof that the subject message has been delivered to the recipient. It is present if the originator requested proof of delivery. The algorithm involved is applied to an instance of the Proof of Delivery Basis class.

Recipient Certificate
The recipient's certificate. Generated by a trusted source (for example, a CA), it constitutes a verified copy of the recipient's PAEK. It is present if the originator requested proof of delivery and an asymmetric encryption algorithm was used to compute the proof.

6.3.5 Per-recipient Non-delivery Check Basis

An instance of class **Per-recipient Non-delivery Check Basis** provides the basis for enabling a third party (for example, a user or an MTA) to verify the origin of a portion of a per-recipient NDR.

The attributes specific to this class are listed in the following table.

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
Non-delivery Diagnostic	Enum (Diagnostic)	-	0-1	-	1988
Non-delivery Reason	Enum (Reason)	-	1	-	1988

Table 6-5 Attributes Specific to Per-recipient Non-delivery Check Basis

Non-delivery Diagnostic
Why in detail the subject message was not, or would not, have been conveyed to the recipient. For its defined values, see Section 5.4.5 on page 144.

Non-delivery Reason
Identifies the factor that prevented, or would have prevented, the subject message from being conveyed to the recipient. For its defined values, see Section 5.4.10 on page 148.

6.3.6 Proof of Delivery Basis

An instance of class **Proof of Delivery Basis** provides the basis for enabling the originator of a message to prove that it has been delivered to a particular recipient. If an asymmetric encryption algorithm is used in the computation, the delivery proof also may provide for non-repudiation of delivery. If a symmetric encryption algorithm is used, the proof may do so only if the security policy in force provides for the involvement of a third party acting as a notary.

The algorithm used to compute the delivery proof may be either symmetric or asymmetric. If the former, the proof is computed by the recipient and validated by the originator using a single key distributed by unspecified means. If the latter, the proof is computed by the recipient using its secret key, and validated by the originator using the associated public key which it can derive from the Recipient Certificate attribute of the per-recipient DR that bears the proof.

The attributes specific to this class are listed in the following table.

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
Actual Recipient Name	Object (OR Name)	-	1	-	1988
Algorithm	Object (Algorithm)	-	1	-	1988
Content	String (Octet)	-	1	-	1988
Content Identifier	String (Printable)	1-16	0-1	-	1988
Delivery Time	String (UTC Time)	0-17	1	-	1988
Originally Intended Recip	Object (OR Name)	-	0-1	-	1988
Security Label	Object (Security Label)	-	0-1	-	1988

Table 6-6 Attributes Specific to Proof of Delivery Basis

Actual Recipient Name
The O/R name of the recipient to which the message, whose delivery is being proved, was delivered. (The recipient may be an alternate recipient.)

Algorithm
Identifies the algorithm used to compute the proof of delivery.

Content
The like-named attribute of the message whose delivery is being proved.

Content Identifier
The like-named attribute, if any, of the message whose delivery is being proved.

Delivery Time
The date and time at which the message, whose delivery is being proved, was delivered to the recipient to which the proof applies.

Originally Intended Recip
The O/R name of the originally intended recipient. This attribute is present if, and only if, the Actual Recipient Name attribute denotes an alternate recipient.

Security Label
The like-named attribute, if any, of the message whose delivery is being proved.

6.3.7 Proof of Submission Basis

An instance of class **Proof of Submission Basis** provides the basis for enabling the originator of a message to prove that it has submitted the message. If an asymmetric encryption algorithm is used in the computation, the submission proof also may provide for non-repudiation of submission. If a symmetric encryption algorithm is used, the proof may do so only if the security policy in force provides for the involvement of a third party acting as a notary.

The algorithm used to compute the submission proof may be either symmetric or asymmetric. If the former, the proof is computed by the originating MTA and validated by the originator using a single key distributed by unspecified means. If the latter, the proof is computed by the originating MTA using its secret key, and validated by the originator using the associated public key which can be derived from the MTA Certificate attribute of the Submission Results that bears the proof.

The attributes specific to this class are listed in the following table.

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
Algorithm	Object (Algorithm)	-	1	-	1988
Message	Object (Submitted Message)	-	1	-	1988
MTS Identifier	Object (MTS Identifier)	-	1	-	1988
Submission Time	String (UTC Time)	0-17	1	-	1988

Table 6-7 Attributes Specific to Proof of Submission Basis

Algorithm
Identifies the algorithm used to compute the proof of submission.

Message
The message whose submission is being proved.

MTS Identifier
The MTS identifier that the MTS assigned to the message whose submission is being proved.

Submission Time
The date and time at which the message whose submission is being proved was submitted.

6.3.8 MT Secret Data

An instance of class **MT Secret Data** provides the basis upon which a token protects secret security-related information. If the Integrity Check attribute is present, the token provides for non-repudiation of the content of the message with which the token is associated. If the Security Label attribute is present also, the token provides proof of the association between the content and the message's security label.

The attributes specific to this class are listed in the following table.

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
Confidentiality Key	String (Bit)	-	0-1	-	1988
Integrity Check	Object (Algorithm and Result)	-	0-1[1]	-	1988
Integrity Key	String (Bit)	-	0-1	-	1988
Message Sequence Number	Integer	-	0-1	-	1988
Security Label	Object (Security Label)	-	0-1[1]	-	1988

[1]If either of these attributes is present, the other must be also if their association is to be maintained.

Table 6-8 Attributes Specific to MT Secret Data

Confidentiality Key
The symmetric encryption key, if any, that the originator of the message that bears the token uses to encrypt its content and that the recipient uses to decrypt the content.

Integrity Check
A means by which any recipient of the message that bears the token can verify that its content is unchanged. The algorithm involved is applied to an instance of the Integrity Check Basis class. This attribute is present at the option of the token's originator.

Integrity Key
The symmetric encryption key, if any, that the originator of the message that bears the token uses to compute the Integrity Check attribute's value and that the recipient uses to verify it.

Message Sequence Number
The like-named attribute, if any, of the message that bears the token. It is included in the token's Secret Information attribute, not Public Information attribute, if its confidentiality is required.

Security Label
The like-named attribute, if any, of the message that bears the token. It is included in the token's Secret Information attribute, not Public Information attribute, if its confidentiality is required.

6.3.9 Token Secret Data

This is an abstract class, providing for future expansion of other token types.

The attributes specific to this class are none.

6.4 Syntax Definitions

There are no SM enumeration syntaxes, that is, syntaxes in the Enumeration group that are specific to SM.

6.5 Declaration Summary

This section lists the declarations that define the portion of the C interface that deals with the SM package.

The declarations, as assembled here, constitute the contents of a header file to be made accessible to client programmers. The header file is **<xsmp.h>**. The symbols the declarations define are the only SM symbols the service makes visible to the client.

Note: The identifier for the variable name of type OM_STRING of a class in the Secure Messaging package can usually be derived using the name of the class, preceded by "SM_C_", and replacing a blank space with an underscore. To be in line with the ANSI C language limitation, some words in the class names are excepted and are abbreviated as below:

RECIPIENT	is abbreviated as	RECIP
NON_DEL		NO_DEL
BASIS		is ignored

```
/* BEGIN SM PORTION OF INTERFACE */

/* SYMBOLIC CONSTANTS */

/* Class */

#define OMP_O_SM_C_INTEGRITY_CHECK        "\x2A\x86\x48\xCE\x22\x0C\x00"
#define OMP_O_SM_C_ORIGIN_CHECK           "\x2A\x86\x48\xCE\x22\x0C\x01"
#define OMP_O_SM_C_PER_RECIP_CHK          "\x2A\x86\x48\xCE\x22\x0C\x02"
#define OMP_O_SM_C_PER_RECIP_DEL_CHK      "\x2A\x86\x48\xCE\x22\x0C\x03"
#define OMP_O_SM_C_PER_RECIP_NO_DEL_CHK   "\x2A\x86\x48\xCE\x22\x0C\x04"
#define OMP_O_SM_C_PROOF_OF_DELIVERY      "\x2A\x86\x48\xCE\x22\x0C\x05"
#define OMP_O_SM_C_PROOF_OF_SUBMISSION    "\x2A\x86\x48\xCE\x22\x0C\x06"
#define OMP_O_SM_C_MT_SECRET_DATA         "\x2A\x86\x48\xCE\x22\x0C\x08"
#define OMP_O_SM_C_TOKEN_SECRET_DATA      "\x2A\x86\x48\xCE\x22\x0C\x09"


/* Type */


#define SM_ACTUAL_RECIPIENT_NAME         ( (OM_type) 400 )
#define SM_ALGORITHM                     ( (OM_type) 401 )
#define SM_CONFIDENTIALITY_KEY           ( (OM_type) 402 )
#define SM_CONTENT                       ( (OM_type) 403 )
#define SM_CONTENT_IDENTIFIER            ( (OM_type) 404 )
#define SM_DELIVERY_POINT                ( (OM_type) 405 )
#define SM_DELIVERY_TIME                 ( (OM_type) 406 )
#define SM_INTEGRITY_CHECK               ( (OM_type) 407 )
#define SM_INTEGRITY_KEY                 ( (OM_type) 408 )
#define SM_INTENDED_RECIPIENT_NAME       ( (OM_type) 409 )
#define SM_MESSAGE                       ( (OM_type) 410 )
#define SM_MESSAGE_SEQUENCE_NUMBER       ( (OM_type) 411 )
#define SM_MTS_IDENTIFIER                ( (OM_type) 412 )
```

```
#define SM_NON_DELIVERY_DIAGNOSTIC       ( (OM_type) 413 )
#define SM_NON_DELIVERY_REASON           ( (OM_type) 414 )
#define SM_PER_RECIPIENT_CHECKS          ( (OM_type) 415 )
#define SM_PROOF_OF_DELIVERY             ( (OM_type) 416 )
#define SM_PROOF_OF_DELIVERY_REQUESTED   ( (OM_type) 417 )
#define SM_RECIPIENT_CERTIFICATE         ( (OM_type) 418 )
#define SM_SECURITY_LABEL                ( (OM_type) 419 )
#define SM_SUBMISSION_TIME               ( (OM_type) 420 )

/* Value Length */

#define SM_VL_DELIVERY_TIME              ( (OM_value_length) 17 )
#define SM_VL_SUBMISSION_TIME            ( (OM_value_length) 17 )

/* Value Number */

#define SM_VN_RECIPIENT_CHECK            ( (OM_uint32) 32767 )


/* END SM PORTION OF INTERFACE */
```

Appendix A

Runtime Binding

This appendix describes how, in the context of selected operating systems, the C implementation of a client may be bound at run time to the C implementation of the service.

A.1 OS/2

Binding of client applications to a service implementation at runtime under OS/2 is accomplished through the use of Dynamic Link Libraries (DLLs). Each interface of a service implementation should be presented to the client as a separate DLL. The service API functions for each library are specified as IMPORTS in the client application's definition (.DEF) file and declared as externals in the client code. This allows the client application to be compiled and linked in absence of the actual service libraries. The OS/2 kernel recognises these unresolved external references in the client application when it is executed and loads the appropriate service DLL.

A.1.1 Service Provider Requirements

Each service interface should be implemented as a separate DLL. Furthermore, different vendor's service implementations must adhere to a consistent DLL naming convention so that client applications may include the service DLL name in their definition files.

Specifically, the Message Access interface library is named maxapia.dll and the Message Transfer interface library is named mtxapia.dll. Other related interfaces not specifically described by this document should use the same convention, for example, the Directory Service interface library should be named dsxapia.dll.

All API functions exposed to the client should use Pascal calling conventions (function parameters are pushed left to right and the called routine clears the stack). This is achieved by declaring each API function within an interface DLL to be far Pascal. Other functions, which are used only within a particular library and not exposed to a client application, may use any calling convention.

Service provider's are free to specify any parameters in definition files associated with the interface libraries as they see fit subject to these restrictions, for example, a service provider may decide to specify an initialisation routine to be executed when a service library is first loaded (the LIBRARY INITINSTANCE directive).

A.1.2 Client Application Requirements

The client application writer must specify a definition file for the client application program and enumerate the API function routines used by the application in the IMPORTS section of the file. The client application will also need to declare each imported API function as using Pascal calling conventions by declaring them as external far Pascal. An example of a definition file (client.def) used by a client application utilising the Message Access and Object Management APIs is given below.

```
NAME CLIENT
IMPORTS
        MAXAPIA.MA_CANCEL_SUBMISSION
        MAXAPIA.MA_CLOSE
        MAXAPIA.MA_FINISH_DELIVERY
        MAXAPIA.MA_FINISH_RETRIEVAL
        MAXAPIA.MA_OPEN
        MAXAPIA.MA_SIZE
        MAXAPIA.MA_START_DELIVERY
        MAXAPIA.MA_START_RETRIEVAL
        MAXAPIA.MA_SUBMIT
        MAXAPIA.MA_WAIT
```

Each service DLL used by a client application must be placed in a directory specified in the LIBPATH directive in the OS/2 config.sys file.

A.2 UNIX System V Release 4.0

Runtime binding in UNIX System V Release 4.0 is accomplished through the use of dynamic shared libraries. Each API must be implemented as a dynamic shared library in the /usr/lib/XAPI directory. Specifically, the functions that make up the MA API will be placed in the /usr/lib/XAPI/libMA.so library, and the MT functions in the /usr/lib/XAPI/libMT.so library. Each dynamic shared library must be implemented to comply with the **UNIX System V Release 4.0 System V Application Binary Interface (ABI) Specification**. There are no additional constraints on the implementation of these libraries. The libraries should, however, be well designed in order to avoid potential performance degradation that can be caused by the use of shared libraries. The **UNIX System V Release 4.0 Programmer's Guide: ANSI C and Programming Support Tools** provides information on how to design shared libraries.

All client applications must be compiled to use the functions in these dynamic libraries. Details of how to link to dynamic shared libraries can be found in the **UNIX System V Release 4.0 Programmer's Guide: ANSI C and Programming Support Tools**. In particular, the client application that uses the APIs may be compiled using absolute path names for the API shared libraries or using the **-L/usr/lib/XAPI** option of the **cc** command.

Both service and client must be compiled with header files that include consistent binary bindings for all values passed between the client and service and consistent definitions of the library interface as specified in the main body of this document.

Appendix B

An Introduction to X.400

This appendix introduces X.400 and its terms and abbreviations, for the information of readers who may be unfamiliar with X.400.

B.1 Background

In 1979 the International Federation for Information Processing (IFIP) organised Working Group (WG) 6.5 to promote both discussion and interchange of information concerning the requirements for, and technology of, computer-based message systems (CBMSs). IFIP is a part of the United Nations Educational, Scientific and Cultural Organisation (UNESCO). IFIP WG 6.5 established many of the basic concepts and terminology used today within the messaging industry.

The requirements for computer-based messaging that IFIP WG 6.5 identified were perceived to be international in nature and to require the support of suitable data communication standards. For these reasons, another international organisation began related work. In 1981 the International Telegraph and Telephone Consultative Committee (CCITT) formed a working, or *rapporteur*, group designated Q5/VII. CCITT is part of the International Telecommunications Union (ITU), which in turn, like UNESCO, is a part of the United Nations. Q5/VII developed, and in 1984 CCITT ratified, standards for CBMS, or *message handling system*, interconnection.

The series of standards for message handling that CCITT ratified in 1984 is known as X.400 (1984). There are eight standards (the CCITT calls them *Recommendations*) in the series: X.400 itself, X.401, X.408, X.409, X.410, X.411, X.420 and X.430. All of the Recommendations that CCITT ratified in 1984, whether related to message handling or not, are bound with red covers. For this reason the 1984 edition of X.400 is sometimes referred to as the *Red Book*.

In 1985 the CCITT formed a new rapporteur group which it designated Q33/VII. This group developed, and in 1988 CCITT ratified, revised standards for message handling. The revised standards repair defects in the original standards and extend their scope to include aspects of message handling beyond those covered initially. Editorially, the new Recommendations represent a complete replacement and major extension of the original ones. Additionally, some of the specifications included in X.400 (1984) were removed from the X.400 series and placed in other Recommendations because of their broad applicability. Most of these more general specifications now reside in the X.200 series of Recommendations for Open Systems Interconnection (OSI).

The series of Recommendations for message handling that CCITT ratified in 1988 is known as X.400 (1988). There are nine Recommendations in the new series: X.400, X.402, X.403, X.407, X.408, X.411, X.413, X.419 and X.420. Because the 1988 Recommendations are bound with blue covers, the 1988 edition of X.400 is sometimes referred to as the *Blue Book*.

Since its ratification, X.400 (1984) has been supplemented by the work of other standards and standards-related groups. They, for example, imposed upper bounds, or *pragmatic constraints*, upon the lengths of certain variable-length protocol fields, constraints that were subsequently incorporated in X.400 (1988). Among these *functional standardisation* groups are the National Institute of Standards and Technology (NIST), formerly the National Bureau of Standards (NBS), the European Conference of Postal and Telecommunications Administrations (CEPT), and the European Committee for Electrotechnical Standardisation (CEN/CENELEC). The objective of the functional standardisation groups has been to help ensure the interoperability of X.400 implementations. Although the *functional profiles* they published differ from one another in some

respects, a system based upon one can successfully interwork with a system based upon another.

B.2 Functional Model

IFIP began, and CCITT completed, a functional model for a *message handling system* (MHS). According to that model, the MHS comprises a single *message transfer system* (MTS) which enables store-and-forward communication between any number of *users*. Each user is assisted by a *user agent* (UA). The MTS comprises one or more interconnected *message transfer agents* (MTAs). The MTS can be likened to a postal system, the MTAs to individual post offices within it.

The service the MTS offers is application-independent. Each *message* the MTS transports has two parts, an envelope and its content. The *envelope* comprises the information that the MTS needs to properly *transfer* and eventually *deliver* the message to the UAs of the users who are its intended *recipients*. Most of this information is supplied by the message's *originator* when that user's UA *submits* the message to the MTS. The message's *content* is arbitrary binary data. Thus the MTS can transport many kinds of information, and thus can support many store-and-forward applications which differ with respect to the nature of the information that users exchange.

The service the MTS offers is also reliable. The originator can request that the MTS return a *delivery report* (DR) when delivery is accomplished. It can also ask the MTS to return a *non-delivery report* (NDR) should it find itself, for any reason, unable to deliver a message. A prospective originator can also test the likelihood of a message's delivery by submitting a probe. A " *probe* comprises an envelope and a description of a message content, rather than the content itself. The MTS transfers the probe up to the point at which the message would have been either delivered or deemed undeliverable. The MTS then discards the probe and returns a DR or an NDR.

Besides standardising various aspects of message transfer in general, X.400 standardises one particular application of the MTS which it calls *interpersonal messaging*. In this application, the content of a message comprises either an *interpersonal message* (IPM) or an *interpersonal notification* (IPN). An IPM can be likened to an electronic memo, an IPN to a high-level acknowledgement of a memo's *receipt* by one of its intended recipients. An IPM has two parts, a *heading* and a *body*, whose purposes are similar to those of the like-named parts of an office memo. The body itself has one or more parts which may vary in nature. The first part, for example, may be text while the second and third parts are G3 facsimile images.

B.3 Application Protocols

X.400 standardises data communication protocols that make it possible to physically realise the MHS functional model by means of separate computer systems interconnected by means of one or more networks. Because the protocols are standardised, the computer systems can be supplied by any number of manufacturers and managed by any number of organisations, public and private.

X.400 (1984) standardises the following principal application-level protocols, relying upon lower-level OSI protocols for its basic data communication needs.

Message Transfer Protocol (P1)
Standardises the transfer of messages and thus the interaction of MTAs.

Message Submission and Delivery Protocol (P3)
Standardises the submission and delivery of messages and thus the interaction of UAs and MTAs.

Interpersonal Messaging Protocol (P2)
Standardises the format of IPMs and IPNs and thus the interaction of UAs, by means of the MTS, in interpersonal messaging.

Major aspects of P1 and P3 are specifications for the syntax and semantics of the fields that make up a message's envelope, and the syntax and semantics of the information objects, called *O/R names*, which appear on the envelope, by means of which individual users are addressed. An O/R name comprises a collection of *attributes* which together identify and locate a user.

P1, P2 and P3 are defined using a high-level specification language called *Abstract Syntax Notation One (ASN.1)*. They are encoded as binary data using a general-purpose encoding regime associated with ASN.1 and called ASN.1's *Basic Encoding Rules (BER)*. The BER dictate a type-length-value (TLV) approach in which each piece of information is self-identifying as to both its type and its length. First appearing as part of X.400 (1984), ASN.1 and BER are among the broadly applicable specifications since moved to the X.200-series.

The MA interface offers the functionality of P3 and P2, the MT interface the functionality of P1 and P2.

B.4 Management Domains

The creation and management of a global MHS is necessarily a collaborative effort involving numerous organisations worldwide. The collection of message handling computer systems operated by one organisation is called a *management domain* (MD). It encompasses one or more MTAs and zero or more UAs. X.400 distinguishes between two kinds of MD. An *administration management domain* (ADMD) comprises equipment operated as a public service by a telecommunication service provider. A *private management domain* (PRMD), on the other hand, comprises equipment operated by any organisation to meet the message handling needs of its employees or members. Thus an ADMD is a public messaging service, a PRMD an in-house messaging system.

The X.400 Recommendations describe ADMDs as being hierarchically organised. ADMDs are connected to one another so as to form a global messaging backbone network. Each PRMD is connected to an ADMD and thus attached to the backbone. The O/R names assigned to users reflect their locations within the hierarchy so that the MTS can efficiently route messages to them.

Appendix C

Gateway Considerations

This appendix identifies some of the issues associated with designing a mail system gateway thast is capable of making use of the MT interface that this document defines.

Every gateway carries out much the same tasks. Exactly how it does so, however, depends in detail upon both the functionality and the architecture of the mail system to which the gateway provides access. Conceptually, the gateway stands between two different message handling universes, that of X.400 and that of the local mail system. The gateway must transfer messages, probes and reports between these two universes. In so doing, it must cope with the differences between the universes, sometimes making compromises of functionality in the process.

Among the tasks that a mail system gateway performs are those discussed below. The list is exemplary not definitive. The approaches mentioned do not necessarily ensure conformance to the X.400 standard, and conformance may be important to a product manufacturer.

A particular gateway is described in **Request for Comment (RFC) 987**, **Mapping Between X.400 and RFC 822**. This paper provides further insight into the issues of, and approaches to, gateway design. For a nominal fee, the paper can be obtained from the following source:

SRI International,
Network Information Systems Center,
333, Ravenswood Avenue, EJ291,
Menlo Park, CA 94025,
U.S.A.

Tel: +1 (415) 859 3695

C.1 Address Translation

X.400 requires that every user of the local mail system is assigned an O/R name. This enables him to receive messages from remote users. X.400 further requires that every local user is enabled to input (as well as display), directly or indirectly, any O/R name that a remote mail system may assign to one of its users. This enables him to send messages to remote users.

The first of the two requirements above typically is easy to satisfy, the second is more difficult. The O/R naming rules are sufficiently rich that the native address of a local user usually can be represented directly as an O/R name. However, the mail system's user interface typically must be modified, sometimes extensively, to enable the input (and display) of O/R names of other forms.

The gateway may translate between local and X.400 addresses however it wishes. The mapping may be accomplished algorithmically (as above) or using a directory. However, the latter approach may be deemed inadequate for remote users. To require that the O/R name of a remote user is registered in such a directory before local users can communicate with him may be too constraining.

C.2 Feature Translation

X.400 defines various *elements of service* (for example, priority delivery) that constitute individual mail system features. Some it declares *basic* or *essential*, meaning that every mail system is to have them. Others it declares *additional*, meaning that some mail systems have them while others may not.

When it transfers a message, probe or report, the gateway translates features of the local mail system to or from features defined by X.400. A particular feature may be defined locally but not by X.400, defined by X.400 but not locally, or defined by both. The latter features pose no particular problem. The gateway may translate other features to or from roughly (but not precisely) equivalent features, act as if the features had not been requested, or declare the message, probe or report unprocessable and (except for a report) issue an NDR.

C.3 Feature Relocation

X.400 assigns specific tasks to specific functional entities in its functional model. The MTA nearest a UA, for example, is to issue any DRs provoked by messages delivered to that UA.

The architecture of the local mail system may necessitate that the gateway performs tasks that, strictly speaking, should be performed within the local mail system.

C.4 Representational Translation

X.400 defines the components of messages, probes and reports using data types defined by ASN.1. For character data, it dictates the use of specific character sets and representations.

When it transfers a message, probe or report, the gateway translates it to or from the format native to the local mail system. In the process the gateway must cope with character set differences. The local character set may be richer than the X.400 character set or vice versa. In such circumstances the gateway may discard characters or, as above, declare the message, probe or report unprocessable and (again, except for a report) issue an NDR.

Appendix D

Differences from IEEE X.400 Standard

The IEEE has published its **Message Handling System (X/Open X.400)** Standard. It is published as IEEE 1224.1-1993 (language independent standard) and IEEE 1327.1-1993 (C binding). Development of this IEEE Standard was based on the X/Open **API to Electronic Mail (X.400)** CAE Specification (November 1991).

The intent of this **Issue 2** of X/Open's **API to Electronic Mail (X.400)** CAE Specification is to align with the corresponding IEEE OM Standard wherever possible. However, there are a few instances where full alignment between the IEEE 1224.1/1327.1 Standard and the X/Open X.400 API CAE Specification has not been achieved.

This Appendix identifies the known substantive differences between this X/Open **API to Electronic Mail (X.400), Issue 2** and the corresponding **IEEE X/Open X.400 Standard**.

D.1 Specific Values of Symbolic Constants

In X/Open X.400, it is mandatory that the implementation supplies the specific `#define` values in the *Declaration Summary* (see Section 3.6 on page 56, Section 4.5 on page 81, Section 5.5 on page 151 and Section 6.5 on page 169). However, in IEEE 1327.1, it is optional that these specific values are supplied.

The effect of this difference is that there may be some IEEE 1224.1-compliant systems that are not X/Open X.400-compliant, but X/Open X.400-compliant systems are not inhibited from being IEEE 1224.1 compliant. Well-behaved applications (that is, those that use the symbolic constants rather than the numerical values) should not be affected by this difference.

D.2 Definition of MH_AC_RECIP_REASSIGNED

In the X/Open X.400 API specification (see Section 5.5 on page 151), the *Enumeration* section of **<xmhp.h>** contains definition

```
#define MH_AC_RECIP_REASSIGNED  ( (OM_enumeration) 2)
```

corresponding to the **recip-reassigned** value of enumeration syntax `Action`. Although the **recip-reassigned** value is defined in IEEE 1224.1, the corresponding constant definition is missing from IEEE 1327.1.

This is an inconsistency in the IEEE standards. It would, however, technically mean that an implementation that conforms to the X/Open X.400 API specification would not conform to IEEE 1327.1.

D.3 Dependency on OM API in MT API

In a previous version of the X/Open X.400 API specification, the description of the Message Transfer function *finish-transfer-in*( ) contained the following:

> The *om_delete*( ) function also unreserves the object to which it is applied.

This would imply that rather than calling *finish-transfer-in*( ), by specifying a single object and moving **equal** to **false**, the user can achieve the same effect by calling *om_delete*( ) with the same object. This means the Object Management (OM) API must interact with the Message Transfer API. The OM API must realise that not only must it delete the object passed to *om_delete*( ), but further actions are required to ensure that the service marks the object as unreserved. This dependency is undesirable.

The above sentence has therefore been removed from the X/Open X.400 description of the Message Transfer function *finish-transfer-in*( ) (see *finish-transfer-in( )* on page 44).

This change represents a difference between the current X/Open X.400 API specification and the API defined by the IEEE 1224.1 standard.

D.4 Usability of Secure Messaging Package

User Agents (UA) wishing to take advantage of the security services offered by the 1988 X.400 standards can do so by using the Secure Messaging Package provided in the X/Open X.400 API.

For example, such UA's may wish to support the Message Origin Authentication Check (MOAC) service. In order to do this the the UA uses an object of the Origin Check Basis class as a parameter to the *om_encode*( ) function to produce an encoding. The encoding is then passed to a crypto function to produce the required value for placing into an Algorithm and Result object or to compare against a value found in a message.

However, classes such as **Origin Check Basis**, **Integrity Check Basis** and **Proof of Delivery Basis** require a mandatory *Content* attribute.

The value of the *Content Octet String* used to compute the MOAC by the originating UA must be the same as that used by the receiving UA when it recomputes the MOAC.

The question arises as to how the UA supplies the *Content Octet String*.

The earlier version of the X/Open X.400 API specification had the following problems which critically affected the usability of the API:

- The IPM and IPN classes were not encodable; the originating UA could not produce a content octet string.

  The encodings obtained were used as the content of a *message*. Since the content is expected to be an *Information Object*, it must be clear that this is what will be generated as the encoding.

- The original value of the content octet string was not passed to the receiving UA; it was decoded into an IPM or IPN object.

  It is not good enough to say that *om_encode*( ) can be used to produce the content octet string, since it may produce a different value to that used by the originating UA because of the flexibility of BER.

As a result, the following changes have been applied to the X/Open X.400 API specification. These changes represent differences between the current X/Open X.400 API and the API defined by the IEEE 1224.1 standard:

1. In the feature table listing Features and their Object Identifiers (see Table 1-2 on page 8), the *General Content* entry and associated note has been added.
2. In Section 4.2 on page 62, superscript "1" has been added to *Interpersonal Message*, *Non-receipt Notification* and *Receipt Notification* in the class hierarchy list, and the accompanying note added:

   [1] These classes are encodable if the SM 88 Package has been negotiated for the session. The encoding shall be of the `Information Object` syntax as specified in X420.
3. In the description of class *Content* (see Section 5.3.6 on page 94), the following paragraph has been added:

   If the General Content feature has been negotiated for the session then a General Content value of this attribute is always returned. This attribute can then be used, for example, as an attribute in objects of classes from the SM88 Package.

D.5 Inconsistency in Different Finish Functions

In a previous version of the X/Open X.400 API specifications, there was an inconsistency in object accessibility after the client finishes with an object in the three interfaces, together with an inconsistency bug in *finish-transfer-in*( ).

The situation was as follows.

- *ma-finish-delivery*( )
  the delivered object might or might not be returned to the delivery queue, based on presence of temporary undeliverability, but its handle is never explicitly stated to be invalidated by the function. The client application *would seem to need* to call *om-delete*( ) explicitly in all cases. There is no statement regarding behavior of the interface if *om-delete*( ) is called on the object before *finish-delivering"* it.
- *mt-finish-retrieval*( )
  the retrieved object might be returned to retrieval queue based on the *remove* argument. There is no statement regarding behavior of the interface if *om-delete*( ) is called on the object before *finish-retrieving* it.

  If *remove* is false, then the function marks it unreserved and puts it back to the queue, and *om-delete*( ) deletes the retrieved object. The application *should not* call *om-delete*( ) on this object later.

  If *remove* is true, the function removes the object from the queue, but is not explicitly specified to *om-delete* the object. The application *would need* to call om-delete later.
- *mt-finish-transfer-in*( ):
  The function description specifies that the object might be returned to the transfer-in queue based on the *remove* argument.

  If *remove* is false, then the function marks it unreserved, puts it back to the queue, and *om-deletes* the object. The application *should not* call om-delete on this object later.

  If *remove* is true, the function removes the object from the queue, but is not explicitly specified to *om-delete* the object. The application *would need* to call *om-delete*( ) later.

  However, the Argument description of the *Object* argument specifies that the object handle is invalidated - that is, in all cases it is invalidated. The description of the *ALL* argument only

indicates conclusion of transfer-in of *all* outstanding transfer-in objects, but nothing about their accessibility.

These problems are resolved in the X/Open X.400 API specification as follows:

- The sentence
  > The *om-delete*( ) function also unreserves the object to which it is applied.

  has been removed from the description of *finish-transfer-in*( ) (see *finish-transfer-in( )* on page 44).

- The following clause has been added to the descriptions of the *ma-finish-delivery*( ), *ma-finish-retrieval*( ) and *ma-finish-transfer-in*( ) functions (see relevant man-pages in Chapter 3):
  > The client should not invoke the OM Delete function on the object prior to calling this function; otherwise the behavior of this function is undefined.

- The following clause has been added to the description of the *ma-finish-delivery*( ):
  > In all cases, the object handle remains valid on return from the function and may be deleted by the client when no longer required.

- The following clause has replaced the text in the description of *ma-finish-retrieval*( ) relating to the disposition of the object on return from the function:
  > In the former case, the service considers that it has transferred responsibility for the object from the service to the client, the object handle remains accessible on return from this function and may be deleted (using the *om-delete*( ) function) when no longer required.
  >
  > In the latter case, the object is returned to the retrieval queue and marked unreserved. The associated object handle is made inaccessible on return from this function and should not be deleted by the client; however, the associated communique or report can be obtained again by invoking a subsequent *start-retrieval*( ) function.

- The following clause has replaced the text in the description of *mt-finish-transfer-in*( ) relating to the disposition of the object on return from the function:
  > In the former case, the service considers that it has transferred responsibility for the objects from the service to the client, the object handles remain accessible on return from this function and may be deleted (using the *om-delete*( ) function) when no longer required.
  >
  > In the latter case, the objects are returned to the input queue and marked unreserved. The associated object handles are made inaccessible on return from this function and should not be deleted by the client; however, the associated communique or report can be obtained again by invoking a subsequent *start-transfer-in*( ) function.

- The sentence
  > The object is made inaccessible.

  has been removed from the Object argument description of *finish-transfer-in*( ).

These changes represent differences between the current X/Open X.400 API and the API defined by the IEEE 1224.1 standard.

D.6 Non-delivery Information for Delivery Queue Users

The API permits a Delivery Queue UA to return non-delivery information for recipients when taking delivery of a message. In a previous version of this API, this only consisted of diagnostic and reason codes.

X.400 permits a printable string containing supplementary information to be passed with these codes to further clarify the problem. This is particularly important for error codes which describe a general problem, such as security-error.

This is an important feature of X.400, so the following changes have been made to the X/Open X.400 API specification, to allow its use. These changes represent differences between the current X/Open X.400 API specification and the API defined by the IEEE 1224.1 standard.

A *Supplementary Info* attribute has been added to the **Local Per-Recipient NDR** class: see Table 5-26 on page 114.

D.7 New Redirection Reason codes

The Draft 1992 *CCITT Recommendation X.411* and the draft amendment soon to be ratified by ISO (*ISO/IEC 10021-4:1990/DAM1*), add a new redirection reason. The X/Open X.400 Implementors Guide Version 9 adds a further one: *alias(4)*.

These redirection reasons are incorporated in X/Open X.400 API specification, but not in the IEEE 1224.1 standard. The X/Open X.400 API specification therefore has the following material that is additional to that in the IEEE 1224.1 and IEEE 1327.1 standards:

- Enumeration syntax Redirection Reason (see Section 5.4.11 on page 149) contains the following values:

  alias
  The intended recipient's preferred address has not been specified. The message has been redirected to that preferred address.

  recipient-directory-substitution
  The communique could not be delivered to the intended recipient via the original O/R Address supplied. However, the O/R Name included a directory name and, from that directory, an alternate O/R Address for the recipient was obtained to which the communique was then delivered.

- <xmhp.h> contains the following additional definitions:

```
#define MH_RR_RECIPIENT_DIRECTORY_SUBS   ((OM_enumeration) 3)
#define MH_RR_ALIAS                      ((OM_enumeration) 4)
```

D.8 SM88 Class Hierarchy

The *Per-recipient Check Basis* class in the SM88 class hierarchy was defined in a previous version of the API as concrete and encodable. It is however not possible to produce a valid encoding of this object since its subclasses provide necessary mandatory attributes. The class is therefore defined as an abstract class in the current X/Open X.400 specification - that is, its name appears in italic type: see Section 6.2 on page 161. This change represents a difference between the current X/Open X.400 API specification and the API defined by the IEEE 1224.1 standard.

D.9 Proof of Submission Description

In the description of the **Proof of Submission Basis** class in a previous version of this API, the last line of the 2nd paragraph read:

> the Reporting MTA Certificate attribute of the DR that bears the proof.

This proof comes back in an instance of the **Submission results** object, NOT the **DR**. In the current X/Open X.400 API specification (see Section 6.3.7 on page 166), this phrase is therefore replaced by:

> Submission Results that bears the proof.

This change represents a difference between the current X/Open X.400 API specification and the API defined by the IEEE 1224.1 standard.

D.10 New DiscardReason code

Technical Corrigendum 5 to ISO/IEC 10021-7, ratified in Autumn 1992 (final text in ISO/IEC JTC 1/SC 18 N 3904), and the X/Open X.400 Implementors Guide Version 9, adds a new DiscardReason: **ipm-deleted(3)**. This is incorporated in the X/Open X.400 API specification (see Section 4.4.2 on page 78), but not in the IEEE 1224.1 standard:

ipm-deleted
The IPM has been deleted before receipt occurred. When a message store is involved deletion occurred before its status became **processed**.

Also, the X/Open X.400 API specification header file **<ximp.h>** contains the following additional definition:

```
#define IM_IPM_DELETED    ((OM_enumeration) 3)
```

D.11 OM_value_number Removed from Header Files

In general, occurrences of the typedef **OM_value_number** have been converted to `OM_uint32` in both the X/Open X.400 API specification and the IEEE 1224.1 standard.

A further occurrence of a value number definition has been found in the definitions for the SM 88 package in Chapter 6 on page 161 (see Section 6.5 on page 169). This definition has been altered to use `OM_uint32`.

This change represents a difference between the X/Open X.400 API specification and the API defined by the IEEE 1224.1 standard.

D.12 Reference to X/Open X.400 Implementor's Guide

The X/Open X.400 API specification has been updated to reference the latest version of this guide, which includes information that supports some of the changes introduced into the specification. The IEEE 1224.1 standard currently references an earlier version of the guide (version 5, 22 February, 1991).

D.13 Value Number for Delivery Report Attribute

The attribute Origin & Expansion History Attribute in Delivery Report corresponds to ASN.1 in the X.411 which was originally defined as:

```
OriginatorAndDLExpansionHistory  ::= SEQUENCE SIZE (0..ub-dl-expansion) OF
```

The X/Open X.400 Implementors Guide Version 8 updated X411 so that the definition is now:

```
OriginatorAndDLExpansionHistory  ::= SEQUENCE SIZE (2..ub-dl-expansion) OF
```

The corresponding value number for this attribute in the class table in the X/Open X.400 API specification (see Section 5.5 on page 151) has been updated accordingly.

D.14 Provision of Report Origin Authentication Check

From X.411 (section 8.3.1.2.1.13), it is up to the reporting MTA to decide whether it provides the Report Origin Authentication Check The wording used there is ''*may be generated*''. A previous version of the X/Open X.400 API specification used the term ''*shall be present*'', which was incorrect. Consequently, the description of the Origin Check attribute of the Delivery Report class (see Section 5.3.14 on page 102) has been updated to read:

> The attribute may be generated by the reporting MTA when a *message (or probe) origin authentication check* was present in the submitted message.

This change represents a difference between the current X/Open X.400 API specification and the API defined by the IEEE 1224.1 standard.

D.15 Mapping Attributes for OR Address onto Session

The class table for OR Address includes a *Note 1* which applies to the attributes *ADMD Name*, *Country Name* and *PRMD Name*.

In a previous version of the API, the note indicated that the default value for each of these attributes is obtained from the value of the attributes with the same names in the current session object.

The class table for the session object contains attributes *ADMD name* and *Country Name* but does not include an attribute *PRMD Name*. Instead it is calls the equivalent attribute *PRMD Identifier*. The *Note 1* for **OR address** has been updated (see Table 5-31 on page 118) to clarify this situation.

This change represents a difference between the current X/Open X.400 API specification and the API defined by the IEEE 1224.1 standard.

D.16 Support of Optional Security Classification Attribute

In the ASN.1 definition for a Security Label in X.411, the field *Security Classification* is defined as optional.

In a previous version of the X/Open X.400 API specification, the corresponding entry in the class table for **Security Label** used the Value Number 1 for the *Security Classification* attribute, but the list of defined values in Section 5.4.14 on page 150 exactly corresponded to those available in the ASN.1. There was no way to represent the case where the Security Classification field is not provided in the ASN.1.

The definition of the *Security Classification* attribute, and the corresponding value number `#define` have been updated to allow the value **not_present** if no security classification is provided.

This change represents a difference between the current X/Open X.400 API specification and the API defined by the IEEE 1224.1 standard.

D.17 Redundant Values for Explicit Conversions

In X.411, the ASN.1 defining Explicit Conversions has had the following value removed: **telex-to-videotex: , videotex-to-telex** (reference X/Open X.400 Implementor's Guide, Version 8, page 32, C62).

The corresponding values in the X/Open X.400 API specification have consequently been removed.

This change represents a difference between the current X/Open X.400 API specification and the API defined by the IEEE 1224.1 standard.

D.18 Handling of Local String Syntax Strings

The X/Open X.400 API specification has been updated to clarify how an application API should handle a Public Object containing string descriptors which have the OM_S_LOCAL_STRING_SYNTAX bit set. The appropriate action is for the API to apply local string translation to the affected string values in the object as it processes it.

This change represents a difference between the current X/Open X.400 API specification and the API defined by the IEEE 1224.1 standard.

Glossary

1988 class
A class all the attributes specific to which are for 1988 alone.

attempted domain
The MD to which a tracing domain attempts but fails to transfer a communique or report.

attempted MTA
The MTA to which a tracing MTA attempts but fails to transfer a communique or report.

C interface
A version of an interface for the variant of C standardised by ANSI.

client instance
A manifestation of the client that shares the client's input and output queues with other instances.

client
Software that uses an interface.

delivery queue
One of two alternative databases by means of which the service conveys objects to the client of the MA interface.

event flag
A Boolean associated with a session and maintained by the service that is used to signal the arrival of objects in the delivery, retrieval or input queue.

feature
A negotiable aspect of an interface.

functional unit
A group of related functions.

generic interface
A version of an interface that is independent of any particular programming language.

input queue
The database by means of which the service conveys objects to the client of the MT interface.

interface
Either the MA interface or the MT interface without distinction, or one or the two in particular.

local environment
The environment of the local MTA.

local MD
The MD of which the local MTA is a part.

local MTA
The MTA that comprises the client and the service in the context of the MT interface.

MA interface
The X.400 Application API.

mail system gateway
Software that uses the MT interface (the client).

MT interface
The X.400 Gateway API.

neighbouring MTA
An MTA with which the service is prepared to establish OSI connections in the context of the MT interface.

network address attributes
The ISDN Number, ISDN Subaddress, Presentation Address and X121 Address attributes specific to the OR Address class.

organisational unit name attributes
The Organisational Unit Name 1, Organisational Unit Name 2, Organisational Unit Name 3 and Organisational Unit Name 4 attributes specific to the OR Address class.

originating domain
The MD at which a communique or report is originated.

output queue
The database by means of which the client of the MT interface conveys objects to the service.

personal name attributes
The Given Name, Initials, Surname and Generation attributes specific to the OR Address class.

reserved
Said of an object in the delivery, retrieval or input queue that the client can access without first removing it from that queue, yet with confidence that no other client can access the object simultaneously.

retrieval queue
One of two alternative databases by means of which the service conveys objects to the client of the MA interface.

service
Software that implements an interface.

source domain
The MD that supplies a piece of trace information.

structured
Said of a postal O/R address that specifies a user's postal address by means of several attributes. Its structure is prescribed in some detail.

subject domain
The MD that contains the MTA embodied by the MT interface's client and service.

subject message
When used in reference to a communique, the communique, if it is a message, or any of the messages denoted by the communique, if it is a probe.

submission queue
The database by means of which the client of the MA interface conveys objects to the service.

tracing domain
The MD that produces an external trace entry.

tracing MTA
The MTA that produces an internal trace entry.

unstructured
Said of a postal O/R address that specifies a user's postal address in a single attribute. Its structure is left largely unspecified.

X.400 Application API
The interface that makes the functionality of the MTS accessible to an MS or a UA, or the functionality of a simple MS accessible to a UA.

X.400 Gateway API
The interface that divides an MTA into two software components, a mail system gateway and an X.400 gateway service.

X.400 gateway service
Software that implements the MT interface (the service).

Index