Solstice™ X.400 Programming Reference Manual

Message Access (MA) Interface
Message Transfer (MT) Interface
Message Store (MS) Interface

SunSoft
A Sun Microsystems, Inc. Business
550 Garcia Avenue
Mountain View, CA 94043
U.S.A.

Part No: 802-5047-10
Revision A, February 1996

Please
Recycle

Adobe PostScript

Contents

Tables

Preface

The *Solstice X.400 Programming Reference Manual* contains a detailed description of the programming interfaces that provide access to an X.400 Message Handling System (MHS). These interfaces adhere to the relevant CAE specifications, developed by X/Open™ in accordance with the X.400 API Association.

This manual includes text derived from the following X/Open documentation with the permission of the X/Open Company Limited:

- API to Electronic Mail (X.400)—CAE Specification
- Message Store API (XMS)—CAE Specification
- EDI Messaging Package—CAE Specification

The text in this manual is not the definitive version of these documents. The definitive versions may be purchased from the X/Open Company Limited, who takes no responsibility for the accuracy of this text. Nothing in this manual shall imply that the text carries the authority of X/Open or that it may be used as the basis for establishing the conformance of any product.

Purpose

This manual defines three application program interfaces (APIs) used in the construction of message handling systems. 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.
- 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.
- A Message Store API that makes the functionality of a message store accessible to a user agent.

The interfaces support Message Access (MA), Message Transfer (MT), Message Store (MS), Interpersonal Messaging (IM), and Electronic Data Interchange (EDI). 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. MS is the storage of messages transferred between the MTS and its users. 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, MT and MS, 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 relevant to all applications, for example, electronic data interchange (EDI).

Who Should Use This Book

This book is intended for system administrators and system programmers engaged in developing applications for the Solstice X.400 Messaging Server and the Solstice X.400 Client Toolkit.

How This Book Is Organized

Chapter 1, "Introduction" provides an overview of the three application programming interfaces that provide access to an X.400 message handling system.

Chapter 2, "Message Access (MA) Interface" describes the programming interface used to convey information between the Message Transfer System (MTS) and User Agents (UAs), and between User Agents (UAs) themselves.

Chapter 3, "Message Transfer (MT) Interface" describes the programming interface used to convey information between the various elements of the Message Transfer System (MTS).

Chapter 4, "Message Store (MS) Interface" describes the programming interface used to interact with a message store.

Chapter 5, "Message Handling Packages" defines the MH 84 Package and the MH 88 Package.

Chapter 6, "Interpersonal Messaging Packages" defines the IM 84 Package and the IM 88 Package.

Chapter 7, "EDI Messaging Package" defines the object classes that constitute the EDI Package.

Chapter 8, "Message Store Package" defines the object classes (except errors) that constitute the Message Store Package. Errors are defined in **Chapter 9, "Message Store Package Errors"**.

Chapter 10, "Message Store General Attributes Package" defines the object classes that constitute the Message Store General Attributes Package.

Chapter 11, "Message Store IM Attributes Package" defines the object classes that constitute the optional Message Store Interpersonal Messaging Attributes Package.

Chapter 12, "Compiling and Linking Applications" includes the compilation statements and include files required to develop applications using the interfaces specified within this manual.

Appendix A, "Header Files" lists the header files that are provided with the Solstice X.400 Messaging Server and the Solstice X.400 Client Toolkit.

Appendix B, "MA and MT Interface Return Codes" contains details of the MH-specific return codes for the Message Access (MA) and Message Transfer (MT) interfaces.

Appendix C, "Referenced Documents" lists the specific documents referenced within this manual.

Glossary is a list of words and phrases found in this book and their definitions.

New Features in This Manual

This manual has the following revisions and additions since the previous version:

- Description of the MS interface functions and packages.
- Addition of class inheritance information. Each section that defines a class includes information on the superclasses from which the class inherits OM attributes.
- Improved online (AnswerBook) navigation. For example, you can click on an object name listed in a Class Hierarchy section to navigate to the description of that object.
- A revised and expanded index.

Terminology and Conventions Used in this Manual

Throughout this manual, *MA interface* denotes the X.400 Application API, *MT interface* denotes the X.400 Gateway API, and *MS interface* denotes the Message Store API. *OM interface* denotes the Object Management API. *XOM specification* refers to the specification defined in the book *Solstice XOM Programming Reference Manual*. The term *service* denotes software that implements an interface. The term *client* denotes software that uses an interface. See the Glossary for definitions of other words and phrases used in this book.

What Typographic Changes Mean

The following table describes the typographic changes used in this book.

Table P-1 Typographic Conventions

Typeface or Symbol	Meaning	Example
`AaBbCc123`	Indicates functions, code, and declaration examples.	The `ma_submit` function.
AaBbCc123	Language-independent names and attribute values. Class names are capitalized, and values are in lower case.	The class **Country Name**. The value **unregistered mail**.
AaBbCc123	Book titles, variables, new words or terms, or words to be emphasized	Read Chapter 6 in *User's Guide*. These are called *class* options. You *must* be root to do this.

The Documentation Set

This document, along with the *Solstice XOM Programming Reference Manual*, is supplied with the following Solstice X.400 products:

- Solstice X.400 Messaging Server
- Solstice X.400 Client Toolkit

Refer to the administrator's guide supplied with the product you purchased for a list of the document set.

Examples

The use of the *Solstice X.400 Programming Reference Manual* is illustrated with programming examples supplied with each product. These are located in subdirectories below `/opt/SUNWconn/mhs/examples`. Each subdirectory includes a `README` file, describing how to use the examples.

Abbreviations

The following abbreviations are used in this document:

Abbreviation	Description
ADMD	Administration Management Domain
ANSI	American National Standards Institute
API	Application Program Interface
ASN.1	Abstract Syntax Notation 1
AU	Access Unit
AVA	Attribute Value Assertion
BER	Basic Encoding Rules
C	Conditional
CA	Certification Authority
CCITT	International Telegraph and Telephone Consultative Committee. Now more frequently referred to as the ITU.
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 organization for Standardization

Abbreviation	**Description**
ITA2	International Alphabet No. 2
ITU	International Telecommunication Union (formerly CCITT)
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
P22	1988 version of the 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

Abbreviation	Description
SFD	Simple Formattable Document
SM	Secure Messaging
UA	User Agent

Introduction 1

This chapter introduces the MT, MA, and MS interfaces and their specifications. Specifically, it:

- Provides an overview of the interfaces
- Introduces the terminology specific to these interfaces and identifies the levels of abstraction and how these levels are related
- Summarizes the way in which implementations differ
- Introduces the features available
- Summarizes the service implementation options
- Gives the conformance requirements imposed upon manufacturers and their products

1.1 Overview

The MA interface uses two persistent queues 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 for subsequent delivery or deliverability verification. The service places inbound messages and reports in a delivery queue for subsequent delivery to the client.

The MT interface uses two persistent queues 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 for subsequent action by the service. The service places inbound messages, probes and reports in the input queue for subsequent action by the client.

Both MA and MT 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 MS interface is designed for operational interactions with a Message Store. It uses information bases to store messages, probes, and reports for subsequent manipulation by the client or service.

The interfaces use facilities provided by the XOM specification. Section 1.5, “Object Management,” on page 1-9 presents some of the important definitions used in the XOM API.

1.2 C Language Binding

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.

1.2.1 C Naming Convention of the MA and MT Interfaces

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 ( ).

Table 1-1 C Naming Conventions of the MA and MT Interfaces

Element Type	**Prefix**	**Case**
Data type	MH_	Upper
Data value	IM_, MH_	Upper
Data value (Class[1])	IM_C_, MH_C_	Upper
Data value (Value Length)	IM_VL_, MH_VL_	Upper
Data value (Value Number)	IM_VN_, MH_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_".

Note – 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.

1.2.2 C Naming Conventions of the MS Interface

The MS interface uses part of the C public namespace for its facilities. All identifiers start with *ms* or *MS*, as shown in Table 1-2.

Table 1-2 C Naming Conventions of the MS Interface

Item	Prefix
functions	*ms_*
error ‘problem’ values	*MS_E_*
OM class names	*MS_C_*
OM value length limits	*MS_VL_*
OM value number limits	*MS_VN_*
other constants	*MS_*
MS attribute types	*MS_A_*
MS IM attribute types	*MS_IM_A_*

Note – The prefixes “msX” and “MSX” are reserved for the proprietary extension of the interface. The prefixes “msP” and “OMP” are also reserved. Client programs should not use any identifier starting with these reserved letters.

The C identifiers are derived from the language-independent names by a systematic process which depends on the kind of name:

- Function names are entirely composed of lower-case letters and prefixed by *ms_*. Thus, `Receive-Result()` becomes *ms_receive_result*().
- C function parameters are derived from the argument and result names by making them entirely lower-case. In addition, the names of results are suffixed by *_return*. Thus, the argument **Selector** becomes selector while the result **Invoke-ID** becomes *invoke_id_return*.
- OM class names are entirely composed of upper-case letters and prefixed by *MS_C_*. Thus, **Fetch-Result** becomes *MS_C_FETCH_RESULT*.
- Enumeration tags are derived from the name of the corresponding OM syntax by prefixing with *MS_*. The case of the letters is left unaltered. Thus, **Enum(Problem)** becomes *MS_Problem*.

- Enumeration constants, as well as the names of OM attributes and all other constants except errors are entirely composed of upper-case letters and prefixed by *MS_*. Thus, **Stored-Messages** becomes *MS_STORED_MESSAGES*.
- Errors are treated as a special case. Constants that are the possible values of the OM attribute **Problem** of a subclass of the OM class **Error** are entirely composed of upper-case letters and are prefixed by *MS_E_*. Thus, **delete-restriction-problem** becomes *MS_E_DELETE_RESTRICTION_PROBLEM*.
- Where names exceed 31 characters in length, the C binding names are abbreviated to be unique within the first 31 characters without affecting the alphabetical ordering of names with the same *MS_* prefix.
- The constants in the *Value Length* and *Value Number* columns of the OM class definition tables are also assigned identifiers. (They have no names in the language-independent specification.) Where the upper limit in one of these columns is not ‘1’ (one), it is given a name consisting of the OM attribute name prefixed by *MS_VL_* for value length or *MS_VN_* for value numbers.
- The sequence of octets for each object identifier is also assigned an identifier for internal use by certain OM macros. These identifiers are all upper-case letters and are prefixed by *OMP_O_*. See the XOM specification for further details on the use of object identifiers.
- Hyphens are converted everywhere to underscores.

1.3 Features

The interfaces enable the client and service to negotiate the use of the various defined features of the MA, MT, and MS interfaces. Features are defined by *functional units* and *packages*. A functional unit (FU) is a group of related functions. A package is a set of OM classes that are grouped together (See Section 1.5.6, “Package,” on page 1-12 for the full definition of a package).

1.3.1 Features of the MA and MT Interfaces

The features defined for the MA interface and the MT interface are identified in Table 1-3. 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.

Table 1-3 Features of the MA and MT Interfaces and their Object Identifiers

Feature	**Symbol**	**Object Identifier (ASN.1) Suffix**[1]
Basic Access FU	basic-access	`mh(5) basic-access(1)`
Submission FU	submission	`mh(5) submission(2)`
Delivery FU	delivery	`mh(5) delivery(3)`
Retrieval FU[2]	retrieval	`mh(5) retrieval(4)`
Basic Transfer FU	basic-transfer	`mh(5) basic-transfer(5)`
Transfer In FU	transfer-in	`mh(5) transfer-in(6)`
Transfer Out FU	transfer-out	`mh(5) transfer-out(7)`
IM 84 Package	im-84	`mh(5) im-84(8)`
IM 88 Package	im-88	`mh(5) im-88(9)`
MH 84 Package	mh-84	`mh(5) mh-84(10)`
MH 88 Package	mh-88	`mh(5) mh-88(11)`
Multiple Delivery[3]	multiple-delivery	`mh(5) multiple(13)`

[1] Each object identifier suffix is appended to:
`{joint-iso-ccitt mhs-motis(6) group(6) white(1) api(2) }`

[2] This functional unit is not supported in the Solstice implementation.

[3] The delivery of a message or report to several local recipients as the result of a single invocation of the `Start Delivery` function. In the absence of this feature, the service confines itself to one recipient for each invocation.

See Section 2.3, “MA Interface Functional Units,” on page 2-9 for a list of the interface functions available within each functional unit.

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.

If the client attempts to use a feature that the service has not made available, the following takes place:

- If the client invokes a function in the Basic Access or Basic Transfer FU, but that FU is unavailable, the service's behavior is undefined.
- If the Basic Access FU is available and the client invokes a function in the Submission, or Delivery 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 sub-objects is an instance of a class that is not in an available package, the no-such-class error arises.

The Basic Access FU and the Basic Transfer FU feature pairs are mutually exclusive for a particular session between the client and the service.

1.3.2 Features of the MS Interface

The features defined for the MS interface are identified in Table 1-4. Optional features are in italics.

Table 1-4 Features of the MS Interface and its Object Identifiers

Feature	**Object Identifier (ASN.1) suffix** [1]
MS Package	`ms-pkg(1)`
MS General Attributes Package	`ms-general-attr-pkg(2)`
MS IM Attributes Package	`ms-im-attr-pkg(3)`
MS FU	`ms-fu(4)`
MS Submission FU	`ms-submission-fu(5)`
MS Administration FU	`ms-administration-fu(6)`
MS Alert FU	`ms-alert-fu(7)`

[1] Each object identifier suffix is appended to:
`{joint-iso-ccitt mhs-motis(6) group(6) white(1) api(2) ms(6) }`

See Section 4.5, “MS Interface Functional Units,” on page 4-11 for a list of the interface functions available within each functional unit.

Use of optional features is negotiated through the *Feature-List* argument of the `Initialize()` function. The service makes an optional feature available to the client only if the client requests it. The client may request a set of features through the *Feature-List* argument to the `Initialize()` function; the MS FU, the MS Package and the MS General Attributes Package are provided by the service as default.

The following occurs if the client attempts to use a feature that the service has not made available:

- If the MS FU is available and the client invokes a function in the other FUs, but that FU is unavailable, the following error arises:

  feature-unavailable [MS_E_FEATURE_UNAVAILABLE].

- After initialising the interface with certain features negotiated, if the client attempts to invoke a function not previously requested, the following error arises:

  feature-not-requested [MS_E_FEATURE_NOT_REQUESTED].

- If the client supplies, as a function argument, an object, one of whose subobjects is an instance of a class not in any of the packages the interface had been initialised with, the following error arises:

  no-such-class [MS_E_NO_SUCH_CLASS].

1.4 Options

A number of aspects of the service’s behavior are implementation-defined. These aspects are summarized 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, or input queue.

Some behavioural aspects of the MS interface are implementation-defined:

- The maximum number of outstanding asynchronous operations.
- Whether an asynchronous function call returns before the operation is submitted to the MS.
- The text and language of error messages.

1.5 Object Management

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

Note – Some terms used (e.g., attribute) are also used in a different context when referring to the Message Store. To avoid confusion, distinct names are used for such terms. Throughout this document, care is taken to distinguish between OM attributes and attributes used with regard to the Message Store. The unqualified term attribute denotes the Message Store construct, whereas the phrase OM attribute denotes the Object Management one.

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 are 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, “Package,” on page 1-12).

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, “OM Class,” on page 1-11).

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, “Descriptor,” on page 1-14).

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, “OM Class,” on page 1-11), 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 lis*t, which can be used directly in a program (see Section 1.5.9, “Descriptor,” on page 1-14).

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. 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**.

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 everywhere else are stated in terms of the number of elements, which are 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, “Message Handling Packages”, defines an OM class called **OR Name**. This OM class has an OM attribute whose value is an OM object of OM class **Name**. 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 *Solstice XOM Programming Reference Manual*.)

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. 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. Refer to the *Solstice XOM Programming Reference Manual* for 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 `DS-Initialize` 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 a message, parts of a message, or function results.

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 `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 recognize 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 `OM-Get`, if required.

Public objects returned by `OM-Get` are read-only and must not be modified in any way.

Message Access (MA) Interface 2

This chapter defines the Message Access (MA) interface. 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.

2.1 MA Interface Architecture

This section provides a functional and architectural overview of the Message Access (MA) interface.

2.1.1 Conceptual Model

As depicted in Figure 2-1, the functions of the MA interface manipulate queues of two kinds (submission and delivery) 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.

Note – The CAE Specification also defines a retrieval queue. This queue is not supported in the current Solstice implementation.

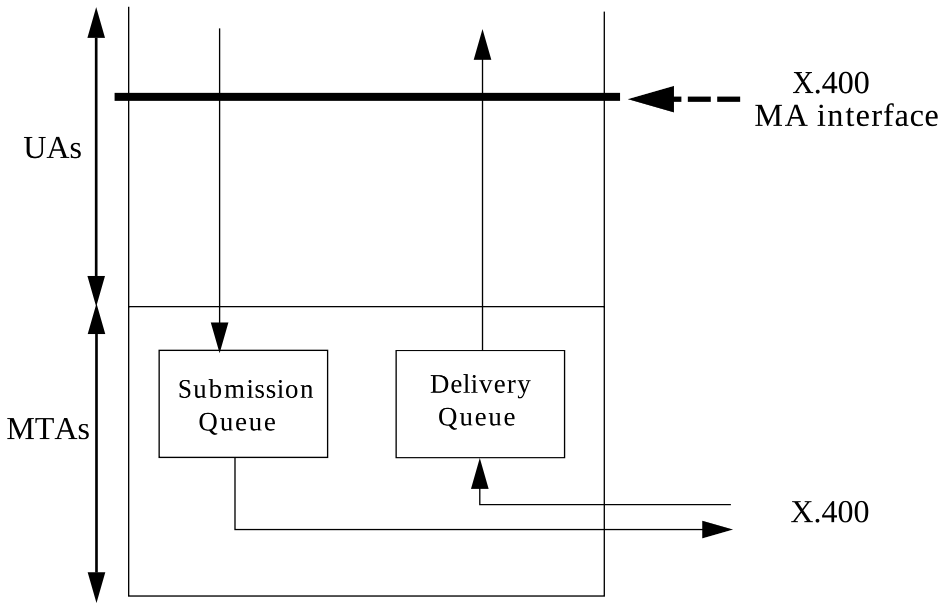

Figure 2-1 MA Interface Queues

The service maintains a submission queue and a delivery queue for each local user, except that a single delivery queue may be shared by a group of local users.

A user's submission queue is added to 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.

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

A user's or user group's delivery queue is added to 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).

The client can access an object in a delivery 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 in which they are made, but rather are terminated by the service. The circumstances that define abnormal termination of a session are system-defined.

2.1.2 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 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 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` 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 Delivery` 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 using the `Size` function, cancel a delivery using the `Finish Delivery` 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.

2.1.3 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).

Besides maintaining the client's submission and delivery 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.

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.

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.
- Convert outbound objects from, and inbound objects to, the client's native format.

Nothing in the 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 protocol mentioned above.

2.2 MA Interface Data Types

This section defines the data types of the MA interface that are specific to MH. The data types of both the generic and C interfaces are specified. The interfaces also include the Boolean, Object, Object Identifier, Private Object, Return Code, String and intermediate data types of the OM interface.

Table 2-1 MA and MT Interface Data Types Specific to MH

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.

Note – These interface declarations are identical to those that apply to the MT interface specified in Chapter 3, “Message Transfer (MT) Interface”.

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.

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, 31) that denotes an interval of time measured in milliseconds.

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, 31) that denotes a number of objects.

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, 231) (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.

2.3 MA Interface Functional Units

MA interface functions are grouped into three FUs (one for basic access and one each for submission and delivery) as indicated in Table 2-2.

Table 2-2 MA Interface Functional Units

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

The OM API is defined in the *Solstice XOM Programming Reference Manual.*

The CAE Specification also defines a Retrieval FU, containing Size, Start Retrieval, Finish Retrieval, and Wait functions.

2.4 MA Interface Functions

Table 2-3 lists the functions of the MA interface defined in this section.

The error return codes associated with this interface are indicated in the error section of each of the function definitions included in this chapter. A detailed description of these error return codes is contained in Appendix B, “MA and MT Interface Return Codes”.

Table 2-3 MA Interface Functions

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.
Open	Establish an MA session.
Size	Determine the size of the delivery queue.
Start Delivery	Begin the delivery of a message or a report.
Submit	Submit a communique.
Wait	Return when an object is available for delivery.

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 of a message or a report is in progress, the service first unsuccessfully finishes that delivery.

Finish Delivery

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

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.

Submit

This function submits a communique (see Chapter 5, “Message Handling Packages”) 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 in the delivery queue to which a session provides access, or when a period of time elapses, whichever occurs first.

The intent of the interface definition is that each function is atomic, that is, that 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. *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, the one that it reports is also 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.

2.4.1 ma_cancel_submission

Cancels a submitted message, with the deferred delivery option.

Synopsis

```
#include <xom.h>
#include <xmh.h>
#include <xmhp.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 is '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 or wrong-value-type.

2.4.2 `ma_close`

Terminates an MA session between the client and the service.

Synopsis

```
#include <xom.h>
#include <xmh.h>
#include <xmhp.h>

OM_return_code
ma_close (
    OM_private_object session
);
```

Description

This function terminates an MA session between the client and the service, and destroys the workspace. If the delivery of a message or a report is in progress in the session, the service first unsuccessfully finishes that 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 is '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 or wrong-class.

2.4.3 ma_finish_delivery

Concludes the delivery in progress in a session.

Synopsis

```
#include <xom.h>
#include <xmh.h>
#include <xmhp.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. (Note that X.400 does not permit an undeliverable report itself to provoke an NDR, but rather requires that the report simply be discarded.) The client may optionally supply delivery confirmations, for users to which 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 defined by the client. 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.

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 is ’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 or wrong-value-type.

2.4.4 ma_finish_retrieval

Concludes the retrieval in progress in a session.

Synopsis

```
#include <xom.h>
#include <xmh.h>
#include <xmhp.h>

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

Description

This function is not implement in the current version of Solstice X.400.

Results (Error)

Feature-unavailable

2.4.5 ma_open

Establishes an MA session between the client and the service.

Synopsis

```
#include <xom.h>
#include <xmh.h>
#include <xmhp.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 client always designates a particular user in the same way. The choice between O/R address and client name is determined by the application. How users are associated with a client name also is determined by the application. 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)
In the Solstice X.400 implementation, this argument is not used and is signalled by the null pointer.

Client Name (String)
In the Solstice X.400 implementation, this argument is mandatory and must correspond to the name of the User Agent as defined in the X.400 configuration. The name of the User Agent is set using `x400tool` as part of the User Agent configuration. Refer to the *Solstice X.400 Messaging Server Administrator's Guide* for detailed instructions.

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 is 'success' or one of the values listed under Errors below.

Feature-List (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 passed as an argument. 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.

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 or wrong-value-type.

2.4.6 ma_size

Determines the number of messages and reports in the delivery queue.

Synopsis

```
#include <mhs/xom.h>
#include <mhs/xmh.h>
#include <xmhp.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 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 is 'success' or one of the values listed under Errors below.

Number (Object Count)
The number of unreserved objects in the delivery 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 or wrong-class.

2.4.7 ma_start_delivery

Begins the delivery of a message or a report to one or more of the users associated with a session.

Synopsis

```
#include <xom.h>
#include <xmh.h>
#include <xmhp.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 `ma_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.

The invocation of this function initiates, but does not complete the delivery; that is, it does not transfer responsibility for the object from the service to the client. This is accomplished by means of the `ma_finish_delivery` function.

Although the `ma_wait` or `ma_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 is ’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 or wrong-class.

2.4.8 ma_start_retrieval

Begins the retrieval of a message or a report.

Synopsis

```
#include <xom.h>
#include <xmh.h>
#include <xmhp.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 is not implement in the current version of Solstice X.400.

Results (Error)

Feature-unavailable

2.4.9 `ma_submit`

Submits a communique**.**

Synopsis

```
#include <xom.h>
#include <xmh.h>
#include <xmhp.h>

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

Description

A communique is a primary information object submitted by users to the MTS. See Section 5.2.51, “Submitted Communique,” on page 5-71 for more information.

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.

Results

Return Code (Return Code)
Whether the function succeeded and, if not, why. It is ’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, not-unreserved, 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 or wrong-value-type.

2.4.10 `ma_wait`

Returns when a message or a report is available for delivery, or when a period of time elapses, whichever occurs first.

Synopsis

```
#include <xom.h>
#include <xmh.h>
#include <xmhp.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 in the delivery 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 queue to which the session provides access, the call returns immediately and signifies itself available.

If there are no unreserved objects on the queue and the specified interval is zero, the call returns immediately and signifies itself 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.

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)
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 is '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 or wrong-class.

2.5 Sequence Of MA Interface Functions

Every client program must first call `ma_open` which returns a workspace. By default, this workspace supports the standard MH Package (see Chapter 5, “Message Handling Packages”. The workspace can be extended to support the optional IM Package (see Chapter 6, “Interpersonal Messaging Packages”), EDIM Package (see Chapter 7, “EDI Messaging Package”), or any vendor extensions. All such packages or other extensions are identified by means of OSI Object Identifiers.

Figure 2-2 shows the order and context in which each MA interface function can be used.

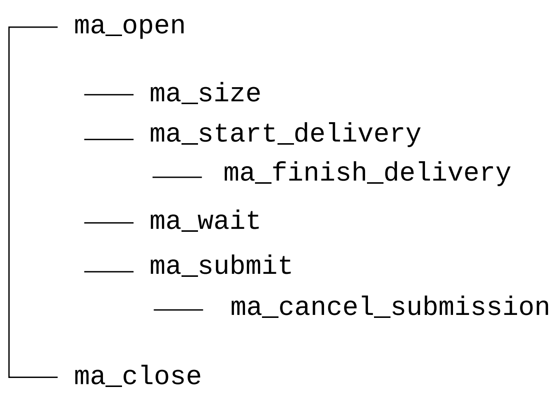

Figure 2-2 Sequence of MA Interface Functions

Once an MA session has been established using the `ma_open` function, the client may call any one of `ma_size`, `ma_start_delivery`, `ma_wait`, or `ma_submit`.

Each session is terminated using `ma_close`. If the MTA is stopped then restarted while a client has an open session, the client must `ma_close` the session before using `ma_open` to re-establish the session.

Message Transfer (MT) Interface 3

This chapter defines the Message Transfer (MT) interface. 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.

You can use the MT interface provided you have Solstice X.400 Messaging Server.

3.1 MT Interface Architecture

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.

3.1.1 Conceptual Model

As depicted in Figure 3-1 on page 3-2, 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.

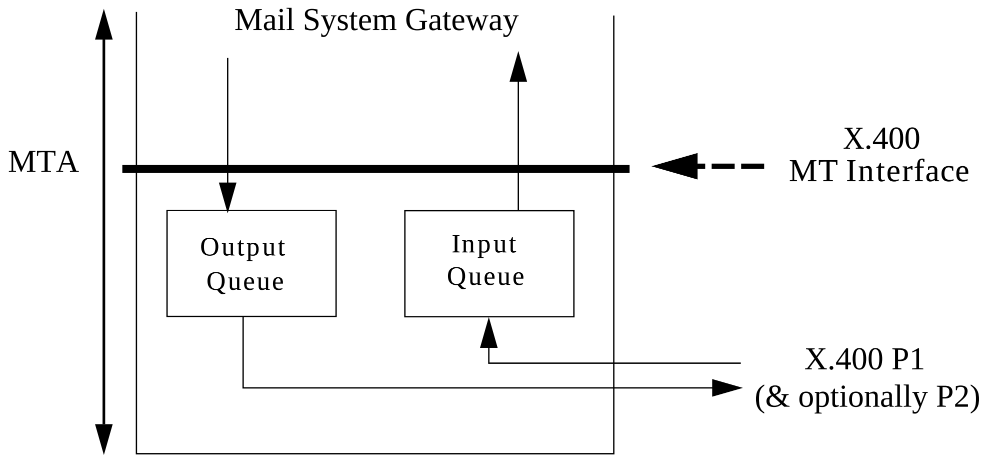

Figure 3-1 Conceptual Model of the X.400 Gateway API

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 in which they are made, but rather are terminated by the service. The circumstances that define abnormal termination of a session are system-defined.

3.1.2 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 `OM 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 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.

3.1.3 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 maximize the tasks the service performs, and thus minimize 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 "neighboring MTAs," in other words, MTAs with which the service is prepared to establish OSI connections.
- Actively establish and terminate OSI connections with, and passively accept connections from, neighboring 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.

Because the client's responsibilities are minimized and the service's responsibilities maximized, the MT interface can be characterized 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 (in other words, the client implementation) by complicating, of necessity, the software that must be written only once (in other words, 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 behavior of an MTA, not dictate the modularity of an MTA implementation.

3.2 MT Interface Data Types

The data types of the MT interface are identical to those that apply to the MA interface. See Section 2.2, "MA Interface Data Types," on page 2-7 for a complete description.

3.3 MT Interface Functional Units

The MT interface functions are grouped into three FUs (one basic and one each for transfer in and transfer out) as indicated in Table 3-1.

Table 3-1 MT Interface Functional Units

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

3.4 MT Interface Functions

This section defines the functions of the MT interface.

The error return codes associated with this interface are indicated in the error section of each of the function definitions included in this chapter. A detailed description of these error return codes is contained in Appendix B, "MA and MT Interface Return Codes".

Table 3-2 MT Interface Functions

Function	**Description**
Close	Terminate an MT session.
Finish Transfer In	Conclude one of the transfers 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.

As indicated in the table, the MT interface comprises a number of functions whose purpose and range of capabilities are summarized 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 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 intent of the interface definition is that each function is atomic; in other words, 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. *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.

3.4.1 `mt_close`

Terminates an MT session between the client and the service.

Synopsis

```
#include <xom.h>
#include <xmh.h>
#include <xmhp.h>

OM_return_code
mt_close (
    OM_private_object session
);
```

Description

This function terminates an MT session between the client and the service and destroys the workspace. 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 is '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 or wrong-class.

3.4.2 mt_finish_transfer_in

Concludes one or all of the transfers in progress in a session.

Synopsis

```
#include <xom.h>
#include <xmh.h>
#include <xmhp.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 objects from the service to the client. In the latter case, the service marks the objects unreserved and makes them inaccessible (in other words, the object handles are made invalid); the associated communique or report, however, can be obtained again in a subsequent `Start Transfer In` function invocation.

The `OM Delete` function also unreserves the object to which it is applied.

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.

An exceptional condition may prevent the client from successfully processing the object. If the client 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 must be (currently) reserved. The object is made inaccessible.

The transfer in of all reserved objects is concluded 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 is ’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 or wrong-class.

3.4.3 mt_open

Establishes an MT session between the client and the service.

Synopsis

```
#include <xom.h>
#include <xmh.h>
#include <xmhp.h>

OM_return_code
mt_open (
    OM_string          client_name,
    OM_string          client_instance_name,
    MH_feature         feature_list[],
    OM_private_objec  *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). In the Solstice X.400 implementation, the client name must correspond to the X.400 P1 User Name as defined in the MTS map. The P1 User Name is set using `x400tool`. Refer to the *Solstice X.400 Messaging Server Administrator's Guide* for detailed instructions.

Client Instance Name (String)
In the Solstice X.400 implementation this argument is ignored and can be passed as the null pointer.

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 is '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 or wrong-class.

3.4.4 `mt_size`

Determines the number of communiques or reports in the input queue.

Synopsis

```
#include <xom.h>
#include <xmh.h>
#include <xmhp.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 is '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 or wrong-class.

3.4.5 `mt_start_transfer_in`

Begins the transfer in of a communique or a report.

Synopsis

```
#include <xom.h>
#include <xmh.h>
#include <xmhp.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.

The invocation of this function initiates, but does not complete, 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 `mt_finish_transfer_in` function.

Although the `mt_wait` or `mt_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.

3.4.6 mt_transfer_out

Adds a communique or a report to the output queue.

Synopsis

```
#include <xom.h>
#include <xmh.h>
#include <xmhp.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.

Results

Return Code (Return Code)
Whether the function succeeded and, if not, why. It is '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 or wrong-value-type.

3.4.7 mt_wait

Returns when a communique or a report is available for transfer in, or when a period of time elapses, whichever occurs first.

Synopsis

```
#include <xom.h>
#include <xmh.h>
#include <xmhp.h>

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

Description

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.

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 in the queue, to which and to how many the service returns early is implementation-defined.

This function is designed to be easily implemented using the event signalling primitives of many operating systems.

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 is ’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 or wrong-class.

3.5 Sequence Of MT Interface Functions

Every client program must first call `mt_open` which returns a workspace. By default, this workspace supports the standard MH Package (see Chapter 5, “Message Handling Packages”. The workspace can be extended to support the optional IM Package (see Chapter 6, “Interpersonal Messaging Packages”), EDIM Package (see Chapter 7, “EDI Messaging Package”), or any vendor extensions. All such packages or other extensions are identified by means of OSI Object Identifiers.

Figure 3-2 shows the order and context in which each MT interface function can be used.

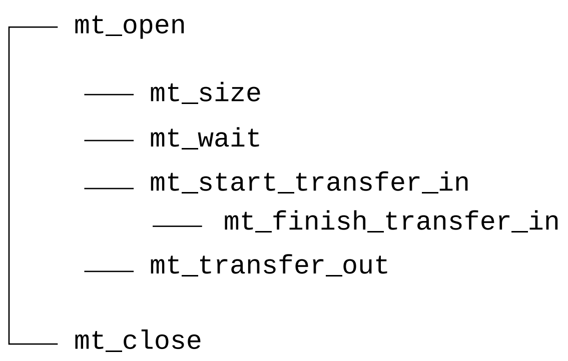

Figure 3-2 Sequence of MT Interface Functions

Once an MT session has been established using the `mt_open` function, the client may call any one of `mt_size`, `mt_start_transfer_in`, `mt_wait`, or `mt_transfer_out`.

Each session is terminated using `mt_close`.

Message Store (MS) Interface 4

This chapter defines the MS interface. It specifies the functions that the service makes available to the client, the data types used by the interface functions and the functional units available for the client to request the usage of subsets of MS functions.

4.1 Message Store Overview

A *Message Store* (MS) acts as an intermediary between a *Message Transfer System* (MTS) and a *User Agent* (UA). The main function of the MS is to accept delivery of messages on behalf of an end-user and to store the messages for subsequent retrieval by the end-user's UA. The MS also provides the UA with the message submission and administration services of the MTS. In this intermediary capacity, the MS can provide a UA with additional functionality compared to direct submission to the MTS.

The indirect submission services offered by the MS includes forwarding of messages stored in the MS.

An MS provides special support for various types of messages, such as interpersonal messages (IPMs).

The P7 protocol gives a UA access to the MS. The MS enables a UA to submit messages, to retrieve messages or parts of messages, and to perform administrative operations such as registration of the MS, the UA, or both. These operations are classified as shown in Table 4-1.

Table 4-1 P7 Protocol Operations

Retrieval	**Indirect Submission**	**Administration**
list	submit message	register
fetch	submit probe	
delete	cancel submission	
register-MS		
alert		
summarize		

In addition, the Solstice implementation provides bind and unbind functions.

The MS stores and maintain a database for each user known as the *stored messages information bases*. This database keeps a copy of all delivered messages and reports.

Note – The CAE Specification for the message store also defines inlog and outlog information bases for logging purposes. The current Solstice implementation does not support these types of information bases.

The elements in an information base of the MS are known as *entries*. An *entry* represents a single object (such as a delivered message) within the information base. An entry consists of a set of *MS attributes*. Each entry is identified by the MS attribute *Sequence-Number* which is unique within an information base and generated when a new entry is created.

An *MS attribute* provides a piece of information about, or derived from, the data to which the entry corresponds. An attribute consists of an *attribute type* which identifies the class of information given by an attribute and its corresponding *attribute value*(s) which are particular instances of that class appearing in the entry. Attributes that can have exactly a single value associated with them are termed *single-valued*, whereas those that can have more than one value at a time are termed *multi-valued* (for example, Other-Recipient-Names). The standards define certain general-purpose attributes, termed *General Attributes*, for the stored messages information bases. Examples of such MS General Attributes types are Entry-Type, Sequence-Number, Originator-Name and Report-Delivery-Envelope. An MS that supports Interpersonal Messaging (IM) will also recognise specific IM attributes and can take appropriate actions on these attributes.

4.2 MS Interface Architecture

The MS interface comprises a number of functions together with many OM classes of OM objects which are used as the arguments and results of the functions. Both the functions and the OM objects are based closely on the Message Store Abstract Service as specified in X.413.

The interface models interactions with the Message Store as service requests made through a number of interface *functions* which take a number of input *arguments*. Each valid request causes an operation within the Message Store which eventually returns a *status* and any *result* from the operation.

All interactions between the user and the MS belong to a session which is represented by an OM object passed as the first argument to most interface functions.

4.2.1 Packages

The MS Package (see Chapter 8, “Message Store Package” and Chapter 9, “Message Store Package Errors”) defines the classes that describe the arguments to the interface functions. The Indirect Submission and Administration operations also require the class definitions of the Message Handling Package (see Chapter 5, “Message Handling Packages”) to be imported.

The MS General Attributes Package (see Chapter 10, “Message Store General Attributes Package”) defines the classes that describe the MS General Attributes. This package is needed to access the General Attributes of entries in the MS.

The MS Interpersonal Messaging Attributes Package (see Chapter 11, “Message Store IM Attributes Package”) defines the classes that describe the MS IM Attributes. If the client wants to view and access the MS entries with individual IM attributes, this package must be supported. If the MS IM Attributes Package is not supported, it is possible to use the interface to retrieve the message content as a whole (encoded in ASN.1) but not the individual IM attributes.

4.2.2 Session

A session provides information about a particular association between a client program and the service. A session identifies the association (or connection) with the MS over which an MS operation is to be sent. A session is commenced by a successful `ms_bind` function and terminated by the `ms_unbind` function. Multiple sessions are allowed, if the maximum number of sessions negotiated through `ms_initialize` is more than one.

A session is described by an OM object of the class Session. An instance of this class is returned by the `ms_bind` function and this is subsequently passed as the first argument to most interface functions.

Detailed specifications of the OM class Session are given in Chapter 8, “Message Store Package”.

4.2.3 Function Arguments

Each interface function takes a number of input arguments (also called input parameters). Session is an argument to several functions.

Each abstract service operation has specific arguments. These are mapped onto corresponding arguments. Arguments that are specific to the MS Abstract Service are grouped together as a single object. For example, one argument to the `ms_fetch` interface function is **Fetch-Argument**.

See Section 4.6, “MS Interface Functions,” on page 4-13 and Section 8.2, “MS Class Definitions,” on page 8-5” for details of the arguments.

All arguments that are OM objects can generally be supplied to the interface functions as public objects (descriptor lists) or as private objects. Private objects must be created in the workspace that was returned by `ms_initialize`. In some cases, constants, representing default or commonly used instances, can be supplied instead of OM objects.

Note – Wherever a function is stated to accept an instance of a particular OM class as the value of an argument, it will also accept an instance of any subclass of that OM class.

4.2.3.1 Attribute

Each MS attribute is represented in the interface by an OM object of the OM class **Attribute**. The type of the MS attribute is represented by an OM attribute, **Attribute-Type**, within the OM object; the values of the MS attribute are expressed as the values of the OM attribute **Attribute-Values**. The form of each value of an MS attribute is determined by the attribute syntax associated with the type of the MS attribute (see Section 10.1, “MS GA Types”, Section 11.1, “MS IM Attribute Types”, and X.413.

The representation of the attribute value depends on the attribute type and is determined as set out below. This lists the manner in which a client program must supply values to the interface. The interface follows the same rules when returning attribute values to the client.

The attribute type and the representation of the corresponding values may be defined in the mandatory MS General Attributes Package defined in Chapter 10, “Message Store General Attributes Package”.

Additional attribute types and their OM representations may be defined in optional packages (for example, the MS Interpersonal Messaging Attributes Package defined in Chapter 11, “Message Store IM Attributes Package”, or future versions of this manual.

In the above cases, attribute values are represented as specified in the class definitions for the packages supported.

Otherwise, the attribute type is not known and the **unavailable-attribute-type** *[MS_E_UNAVAILABLE_ATTRIBUTE_TYPE]* error arises.

Where attribute values have OM syntax String(*), they may be long, segmented strings and the functions `OM-Read()` and `OM-Write()` should be used to access them.

When an attribute value is returned indicating no fields present - for example the ASN.1 contains an empty SET - then the No-Value bit of the attribute value descriptor’s syntax is set, and the descriptor’s value is not present.

4.2.3.2 AVA

An attribute value assertion (AVA) is an assertion about the value of an attribute of an (MS) entry and, in the context of MS, can be true or false. It consists of an attribute type and a single value. Loosely, the AVA is true if one of the values of the given attribute in the entry matches the given value. An AVA is represented in the interface by an instance of the OM class **AVA** which is a subclass of **Attribute** constrained to have precisely one value.

4.2.4 Function Results

All functions return a **Status** (in other words, the function result in the C binding). Some MS functions return a **Result**. In the asynchronous mode, all functions return an **Invoke-ID**, which identifies the particular invocation. (In the C binding, the **Invoke-ID** and **Result** are returned using pointers that are supplied as arguments to the C function.) These three kinds of function results are introduced below.

All OM objects returned by interface functions (results and errors) will be private objects in the workspace returned by `ms_initialize`.

4.2.4.1 Status

Every interface function returns a **Status** value, which is either the constant success *{MS_SUCCESS}* or an error. Errors are represented by private objects whose OM classes are subclasses of Error, unless the interface has not been initialized successfully in which case the constant No-Workspace *{MS_NO_WORKSPACE}* is used. Details of all errors are given in Chapter 9, “Message Store Package Errors”.

Other results of functions are not valid unless the status result has the value **success**.

4.2.4.2 Invoke-ID

All interface functions that can be asynchronously invoked return an **Invoke-ID** which is an integer identifying the particular invocation of the function on a particular session. The **Invoke-ID** is only relevant for asynchronous functions and may be used later to receive the result and status. Asynchronous operations are fully described in Section 4.3, “Synchronous And Asynchronous Operations,” on page 4-8” and the interface functions that can be used to start them are indicated in Section 4.5, “MS Interface Functional Units,” on page 4-11”.

The numerical value returned from a call that successfully invokes an asynchronous operation is guaranteed to be unique among all outstanding operations within a given session. The value is such as could be returned from the Remote Operations Service Element (ROSE) defined in CCITT X.219/X.229 and ISO 9072.

The value of the **Invoke-ID** returned for a synchronous function call is unspecified, as is that for a call that fails to invoke an operation.

4.2.4.3 Result

Certain functions return a result only if they succeed. For unsuccessful outcomes, errors from such functions are reported in the **Status** described below (as are errors from all other operations).

The value of **Result** returned by a function that invokes an asynchronous operation is unspecified. The result of an asynchronous operation is returned by a subsequent call to `ms_receive_result`.

The result is returned in a private object whose OM class is appropriate to the particular function. The format of such results is driven both by the Abstract Service and by the need to provide asynchronous execution of these functions. To simplify processing of asynchronous results, the result of a single function is returned in a single OM object (corresponding to the abstract result defined in the Standards). The components of the result of a function are represented by OM attributes in the **Result** object. All information in the Abstract Service result is made available to the client. The result can be examined using functions provided in the *Solstice XOM Programming Reference Manual*.

Any attribute values in the result are represented as described in Section 4.2.3.1, "Attribute," on page 4-5.

4.3 Synchronous And Asynchronous Operations

The support of asynchronous operations is an optional feature. The support of asynchronous operations is indicated by the value of **max-outstanding-operations**, described below. Implementations that support asynchronous operations may be executed in either the *synchronous mode* or the *asynchronous mode*. Implementations that do not support asynchronous operations can only be executed in the synchronous mode.

The asynchronous mode is chosen by appropriately setting the value of the **Maximum-Outstanding-Operations-Requested** attribute in the argument to `ms_initialize`. If this number is zero, all operations will be performed synchronously; however, if this number is greater than zero, the asynchronous mode is requested.

In *synchronous* mode, all functions wait until the operation is complete before returning. Thus, the thread of control is blocked within the interface after calling a function and it can make use of the result immediately after the function returns. (Note that in a multi-threaded system, only one thread in the process is blocked, and use of asynchronous mode is likely to be rare on such systems. On conventional single-thread process systems, the entire process is blocked; and hence the need for the asynchronous mode).

In *asynchronous* mode, some functions return before the operation is complete. The functions that can be executed asynchronously are indicated in Section 4.6, "MS Interface Functions," on page 4-13. The application is then able to continue with other processing while the operation is being performed by the Message Store; and later can access the result by calling `ms_receive_result`.

An application may initiate several concurrent asynchronous operations on the same session before receiving any of the results, subject to the limit described below. The results from asynchronously executed operations are not guaranteed to be returned in any particular order.

An asynchronous function call returns an **Invoke-ID** which uniquely identifies the function invocation. The (synchronous) function `ms_receive_result` returns an **Invoke-ID** corresponding to an outstanding function invocation and the results of that invocation.

Implementations will also define a limit on the number of asynchronous operations that may be outstanding at any one time on any one session. An asynchronous operation is outstanding from the time the function is called until the result is returned by `ms_receive_result`. The limit is given by the constant:

max-outstanding-operations *{MS_MAX_OUTSTANDING_OPERATIONS}*,

which can be negotiated through the `ms_initialize` function. This limit has the value zero if asynchronous operations are not supported. If the feature is present, it is guaranteed to be at least one, so an application can always use the interface in asynchronous mode. While the maximum number of operations is outstanding, attempts to call further asynchronous operations will report an MS Library-Error (**too-many-operations**).

A synchronous call, other than `ms_receive_result`, may return an MS Library-Error (**mixed-synchronous**), if it is made on a session on which there are any outstanding asynchronous operations. All asynchronous operations should be allowed to terminate and their results should be obtained before making a synchronous call on the same session.

For asynchronous calls, certain forms of errors may be detected and reported immediately by the service. In such instances, the function call returns an error immediately and no outstanding operation is generated, and the value of the **Invoke-ID** is undefined. Errors detected after the asynchronous function call has returned are reported later, in the result of the function retrieved using the `ms_receive_result` function. All errors occurring during a synchronous request are reported when the function returns. Full details of error handling are given in Chapter 9, “Message Store Package Errors”.

Clients should ensure that there are no outstanding asynchronous operations on a session when `ms_unbind` is called on that session. Once `ms_unbind` has been called, there is no way to determine whether any outstanding operations have been completed. No errors or results of any kind will be reported to the client and outstanding calls may be left partially completed. Hence, it is strongly recommended that before closing a session, `ms_receive_result` be called repeatedly until the **Completion-Flag** takes the value of **no-outstanding-operation**.

4.4 MS Interface Data Types

This section defines the data types of the MS interface. The data types of both the generic and C interfaces are specified. The interface also includes the Boolean, Object, Object Identifier, Private Object, Return Code, String and intermediate data types of the OM interface.

Table 4-2 MS Interface Data Types

Data Type	Description
Feature	The features requested for a session.
Invoke-ID	Identifies a particular invocation of an interface function in the asynchronous mode.
Status	Indicates whether a function has succeeded or not; and if not, gives the possible error value.

Name

Feature—type definition for requesting features.

C Declaration

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

Description

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

Name

Invoke-ID—type definition for identifying a particular interface function invocation in the asynchronous mode.

C Declaration

```
typedef OM_sint       MS_invoke_id;
```

Description

A data value of this type is used for identifying a particular interface function invocation in the asynchronous mode.

Name

Status—type definition for indicating the outcome (success or error) of a function.

C Declaration

```
typedef OM_private_object MS_status;
```

Description

A data value of this type is used for indicating the outcome (success or error) of a function.

4.5 MS Interface Functional Units

A client program may request certain features of the Message Store it would like to use for the duration of an API session. Such features may be negotiated in terms of functional units and packages. A *functional unit* (FU) is a collection of related functions. Table 4-3 on page 4-12 shows the interface functions available within each functional unit.

The functions are grouped into four FUs (one basic and one each for submission, administration, and alert) as indicated in Table 4-3. The MS FU is mandatory.

Table 4-3 MS Interface Functional Units

MS FU	**MS Submission FU**	**MS Administration FU**	**MS Alert FU**
Bind	Cancel-Submission	Register	Check-Alert
Delete	Submit		
Fetch			
Initialize			
List			
Receive-Result[1]			
Register-MS			
Shutdown			
Summarize			
Unbind			
Wait			

[1] available only if asynchronous mode is supported

Functional units are requested using the **Feature-List** argument to the `ms_initialize` function. After the interface has been initialized with certain features negotiated, if the client attempts to invoke functions not previously negotiated, an error, **feature-not-requested** *[MS_E_FEATURE_NOT_REQUESTED]*, occurs.

All FUs must be used together with the MS General Attributes Package (see Chapter 10, "Message Store General Attributes Package").

Note – The functions of the Submission FU and the Administration FU require class definitions to be imported from the Message Handling Package. See Chapter 5, "Message Handling Packages".

4.6 MS Interface Functions

The MS interface functions are summarised in Table 4-4 below. Functions that can execute asynchronously are indicated by an 'a' in the table; all other functions always execute synchronously. All errors, including Message Store errors, are returned in **Status** (see Section 4.2.4.1, "Status," on page 4-7).

Table 4-4 MS Interface Functions

	Name	Description
	Bind	Establish a session with the Message Store.
a	**Cancel-Submission**	Cancel a message submitted with the deferred delivery option.
	Check-Alert	Check if the MS has received new entries whose attributes match the criteria previously supplied by the `ms_register_ms` function.
a	**Delete**	Remove selected entries from an information base.
a	**Fetch**	Get information on a specific entry in an information base.
	Initialize	initialize the interface, returning a workspace.
a	**List**	Return selected information for a list of entries of interest from an information base.
	Receive-Result	Retrieve the result of an asynchronously executed operation.
a	**Register**	Modify various parameters held by the MTS regarding delivery of messages to the MS.
a	**Register-MS**	Register or deregister various information with the MS.
	Shutdown	Shut down the interface, discarding the workspace.
a	**Submit**	Submit a communique (message or probe).
a	**Summarize**	Summarise counts of selected entries in an information base.
	Unbind	Terminate a session with the Message Store.
a	**Wait**	Return when a new entry is available in the Message Store for retrieval or when a specified period of time has elapsed, whichever occurs first.

4.6.1 `ms_bind`

Establishes a session with the Message Store.

Synopsis

```
#include <xom.h>
#include <xms.h>
#include <xmsga.h>

MS_status ms_bind (
    OM_object           bind_argument,
    OM_private_object  *bind_result_return,
    OM_private_object  *bound_session_return
);
```

Description

This function establishes a session with the Message Store. It must be called after `ms_initialize` and before any other Message Store interface functions are called.

Arguments

Bind-Argument (Object(Bind-Argument))
Specifies information for establishing a session with the Message Store service provider, together with details of the service required. This comprises:

- Initiator
  specifies the OR-name of the initiator (the UA) of this session (or association) with the MS. In the Solstice X.400 Message Store implementation, this argument must correspond to the name of the User Agent as defined in the X.400 configuration. The name of the User Agent is set using `x400tool` as part of the User Agent configuration. Refer to the *Solstice X.400 Messaging Server Administrator's Guide* for detailed instructions. Note that the OR-name is case sensitive.
- Initiator-Credentials
  specifies the credentials of the initiator for authentication purposes.
- Security-Context
  identifies the security context at which the initiator proposes to operate.

- Fetch-Restrictions
  specifies the restrictions on entries to be returned as result of a `ms_fetch` function. These restrictions prevail until the `ms_unbind` function is called.
- MS-Configuration-Request
  if true, specifies the request to obtain information relating to which auto-actions and optional attributes the MS provides support for. If false, no such request is being made.

Results

Status (Status)
Indicates whether the function succeeded or not.

Bind-Result (Object(Bind-Result))
On successful completion, contains one or more of the following:

- Responder-Credentials
  specifies the credentials of the MS.
- Available-Auto-Actions
  specifies the set of all possible auto-actions that are supported by the MS (not just those requested by the client), if the MS-Configuration-Request was made for the `ms_bind` function.
- Available-Attribute-Types
  specifies the set of all optional MS attribute-types that are supported by the MS, if the MS-Configuration-Request was made for the `ms_bind` function.
- Alert-Indication
  indicates an alert condition has occurred since the last successful Alert-indication.
- Content-Types-Supported
  specifies a set of object-identifiers defining the content-types of which the MS has knowledge, if the MS-Configuration-Request was made for the `ms_bind` function.

Bound-Session (Object(Session))
On successful completion, contains an instance of the Session class describing an association between the client and the service. This value is then used as an input argument, Session, to other functions (for example, `ms_fetch`).

Errors

This function can return a **System-Error** or one of the following **Library-Errors**: bad-argument, too-many-sessions, miscellaneous.

The following Message Store errors may be returned: authentication-error, no-workspace, unacceptable-security-context, unable-to-establish-association.

This function can return a **Communications-Error**.

4.6.2 ms_cancel_submission

Cancels a message submitted with the deferred delivery option.

Synopsis

```
#include <xom.h>
#include <xms.h>
#include <xmsga.h>

MS_status ms_cancel_submission (
    OM_private_object     session,
    OM_object             mts_identifier,
    MS_invoke_id          *invoke_id_return
);
```

Description

This function attempts to cancel the delivery of a message submitted with the deferred delivery option, regardless of the session in which it was submitted.

Arguments

Session (Object(Session))
Specifies the MS session over which this function is performed.

MTS-Identifier (Object(MTS-Identifier))
Refers to the MTS-Identifier assigned to the messages whose delivery is to be cancelled.

Results

Status (Status)
Indicates whether the function succeeded or not, if used synchronously; or whether the function has been initiated, if used asynchronously.

Invoke-ID (Integer)
Specifies the Invoke-ID of the asynchronous operation.

Errors

This function can return a **System-Error** or one of the following **Library-Errors**: bad-argument, bad-session, bad-class, no-such-class, asynchrony-not-supported, feature-unavailable, feature-not-negotiated, out-of-memory, miscellaneous, too-many-operations.

The following Message Store errors may be returned: deferred-delivery-cancellation-rejected, message-submission-identifier-invalid, no-workspace, remote-bind-error.

This function can return a **Communications-Error**.

4.6.3 ms_check_alert

Check if the MS has received new entries whose attributes match the criteria previously supplied by the `ms_register_ms` function.

Synopsis

```
#include <xom.h>
#include <xms.h>
#include <xmsga.h>

MS_status ms_check_alert (
    OM_private_object      session,
    OM_private_object     *check_alert_result_return
);
```

Description

This function is used to check if the MS has received new entries whose attributes match the criteria previously supplied by the `ms_register_ms` function.

Alerts received prior to the session being established are not reported by the `ms_check_alert` function; an indication of these alerts is returned as a result of the `ms_bind` function.

Arguments

Session (Object(Session))
Specifies the MS session over which this function is performed.

Results

Status (Status)
Indicates whether the function succeeded or not.

Result (Object(Check-Alert-Result))
If there are alerts, each element of the result may contain the following:

- Alert-Registration-Identifier
  identifies which of the auto alert registrations resulted in the alert.

- New-Entry
  if present, conveys the information from the new entry which was requested in the auto alert registration parameter; otherwise, is absent when the user did not specify an auto alert registration parameter.

Errors

This function can return a **System-Error** or one of the following **Library-Errors**: bad-session, asynchrony-not-supported, feature-unavailable, feature-not-negotiated, out-of-memory, miscellaneous, too many operations.

The following Message Store errors may be returned: no-workspace, security-error.

This function can return a **Communications-Error**.

4.6.4 `ms_delete`

Removes selected entries from an information base.

Synopsis

```
#include <xom.h>
#include <xms.h>
#include <xmsga.h>

MS_status ms_delete (
    OM_private_object  session,
    OM_object          delete_argument,
    MS_invoke_id       *invoke_id_return
);
```

Description

This function is used to delete selected entries from an information base. A main-entry and all its dependent child-entries may only be deleted together. This is achieved by specifying just the main-entry as an argument. The function will only be successful when operating on those information-bases permitted by the security-context and the security-policy in force.

Arguments

Session (Object(Session))
Specifies the MS session over which this function is performed.

Delete-Argument (Object(Delete-Argument))
Specifies the argument for the `ms_delete` function. It comprises:

- Information-Base-Type
  specifies the repository containing entries for delivered messages and reports. Its value is stored-messages *{MS_STORED_MESSAGES}*
- Items specifies the entries to be deleted.

Results

Status (Status)
Indicates whether the function succeeded or not, if used synchronously; or whether the function has been initiated, if used asynchronously.

Invoke-ID (Integer)
Specifies the Invoke-ID of the asynchronous operation.

Errors

This function can return a **System-Error** or one of the following **Library-Errors**: bad-argument, bad-session, bad-class, no-such-class, asynchrony-not-supported, feature-unavailable, feature-not-negotiated, out-of-memory, miscellaneous, too-many-operations.

The following Message Store errors may be returned: delete-error, invalid-parameter-error, no-workspace, range-error, security-error, sequence-number-error, service-error.

This function can return a **Communications-Error**.

4.6.5 `ms_fetch`

Gets selected information from a specific entry in an information base.

Synopsis

```
#include <xom.h>
#include <xms.h>
#include <xmsga.h>

MS_status ms_fetch (
    OM_private_object   session,
    OM_object           fetch_argument,
    OM_private_object  *fetch_result_return,
    MS_invoke_id       *invoke_id_return
);
```

Description

This function is used to obtain selected information from a specific entry in an information base; alternatively, it is used to obtain the first entry from among several entries of interest, in which case, the sequence numbers of the other selected entries are also returned.

Arguments

Session (Object(Session))
Specifies the MS session over which this function is performed.

Fetch-Argument (Object(Fetch-Argument))
Specifies the argument for the `ms_fetch` function. It comprises:

- Information-Base-Type
  specifies the repository containing entries for delivered messages and reports. Its value is stored-messages *{MS_STORED_MESSAGES}*
- Item
  specifies the selector determining the entry to be fetched.
- Requested-Attributes
  indicates what information from the selected entry is to be returned in the result.

Results

Status (Status)
Indicates whether the function succeeded or not, if used synchronously; or whether the function has been initiated, if used asynchronously.

Result (Object(Fetch-Result))
On successful completion of a synchronous call, contains one or more of the following:

- Entry-Information
  specifies the requested entry-information from the selected entry (unless the function returns without any entries being selected).
- List
  specifies, in the case a search was performed and more than one entry matching the selector were found, a list of sequence-numbers, in ascending order, of these entries.
- Next
  specifies, in the case where the number of entries selected would have been greater if it were not for the limit specified in the selector, the sequence-number for the next entry.

Invoke-ID (Integer)
Specifies the Invoke-ID of the asynchronous operation.

Errors

This function can return a **System-Error** or one of the following **Library-Errors**: bad-argument, bad-session, bad-class, no-such-class, asynchrony-not-supported, feature-unavailable, feature-not-negotiated, out-of-memory, miscellaneous, too-many-operations.

The following Message Store errors may be returned: attribute-error, fetch-restriction-error, invalid-parameter-error, no-workspace, range-error, security-error, sequence-number-error, service-error.

This function can return a **Communications-Error**.

4.6.6 `ms_initialize`

initializes the interface, returning a workspace.

Synopsis

```
#include <xom.h>
#include <xms.h>
#include <xmsga.h>

MS_status ms_initialize (
    MS_feature          feature_list[],
    OM_sint            *max_sessions,
    OM_sint            *max_outstanding_operations,
    OM_workspace       *workspace_return,
);
```

Description

This function initializes the interface, including creating a workspace and making the mandatory features (see Section 1.3.2, "Features of the MS Interface," on page 1-7) available. It must be called before any other Message Store interface functions are called. If it is subsequently called before `ms_shutdown`, the result is unspecified.

Arguments

Feature-List (Feature-List)
Identifies the additional features (FUs and packages) requested by the client. This is an ordered sequence of features, each represented by an object identifier (see Section 1.3.2, "Features of the MS Interface," on page 1-7 for the features defined in this document). 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). Mandatory features are made available even if no features are requested.

Maximum-Sessions (Integer)
Indicates the maximum number of simultaneous MS sessions requested by the client.

Maximum-Outstanding-Operations (Integer)
Indicates the maximum number of outstanding asynchronous operations requested by the client.

Results

Status (Status)
Indicates whether the function succeeded or not.

Feature-List (Boolean-List)
If the function completed successfully, contains an ordered sequence of Boolean values, with the same number of elements as the Feature-List. A value of true indicates that the corresponding feature is part of the interface whereas a value of false 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 *MS_feature*, which is defined as:

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

Maximum-Sessions (Integer)
Gives the maximum number of simultaneous MS sessions requested or the maximum number that can be supported by the service, whichever value is lesser.

Maximum-Outstanding-Operations (Integer)
Gives the maximum number of outstanding asynchronous operations or the maximum number that can be supported by the service, whichever value is lesser. If the service does not support asynchronous operations, then the result returned is zero; otherwise it is a positive integer.

Workspace (Workspace)
On successful completion, contains a handle to a workspace in which OM objects can be created and manipulated. Objects created in this workspace may be used as arguments to other interface functions.

Errors

No-workspace.

4.6.7 `ms_list`

Returns selected information for a list of entries of interest from an information base.

Synopsis

```
#include <xom.h>
#include <xms.h>
#include <xmsga.h>

MS_status ms_list (
    OM_private_object   session,
    OM_object           list_argument,
    OM_private_object  *list_result_return,
    MS_invoke_id       *invoke_id_return
);
```

Description

This function is used to obtain selected information for a list of entries selected from an information base.

Arguments

Session (Object(Session))
Specifies the MS session over which this function is performed.

List-Argument (Object(List-Argument))
Specifies the argument for the `ms_list` function. It comprises:

- Information-Base-Type
  specifies the repository containing entries for delivered messages and reports. Its value is stored-messages *{MS_STORED_MESSAGES}*
- Selector
  determines the entries are to be returned.
- Requested-Attributes
  indicates the information from the selected entry to be returned in the result.

Results

Status (Status)
Indicates whether the function succeeded or not, if used synchronously; or whether the function has been initiated, if used asynchronously.

Result (Object(List-Result))
On successful completion of a synchronous call, contains one or both of the following:

- Next
  specifies the sequence-number of the next entry that would have been selected in the case where the number of entries selected would have been greater if it were not for the limit specified in the Selector.
- Requested
  specifies the requested entry-information from each selected entry (one or more), in ascending order of sequence-number (unless the function returns without any entries being selected).

Invoke-ID (Integer)
The Invoke-ID of the asynchronous operation.

Errors

This function can return a **System-Error** or one of the following **Library-Errors**: bad-argument, bad-session, asynchrony-not-supported, feature-unavailable, feature-not-negotiated, out-of-memory, miscellaneous, too-many-operations.

The following Message Store errors may be returned: attribute-error, invalid-parameter-error, no-workspace, range-error, security-error, sequence-number-error, service-error.

This function can return a **Communications-Error**.

4.6.8 `ms_receive_result`

Retrieves the result of an asynchronously executed operation

Synopsis

```
#include <xom.h>
#include <xms.h>
#include <xmsga.h>

MS_status ms_receive_result (
    OM_private_object   session,
    OM_uint            *completion_flag_return,
    MS_status          *operation_status_return,
    OM_private_object  *result_return,
    MS_invoke_id       *invoke_id_return
);
```

Description

This function is used to retrieve the completed result of some operation previously executed asynchronously.

The function results include two status indications. **Status**, indicates whether this function call itself was successful; it is always returned. **Operation-Status**, is used to return the status of the completed asynchronous operation and is only returned if it exists.

Arguments

Session (Object(Session))
Specifies the MS session over which this function is performed.

Results

Status (Status)
Indicates whether the function succeeded or not.

Completion-Flag (Unsigned-Integer)
Indicates the status of outstanding asynchronous operations and has one of the following values:

- completed-operation *{MS_COMPLETED_OPERATION}* indicates that at least one outstanding asynchronous operation has completed and the result for one is made available.
- outstanding-operation *{MS_OUTSTANDING_OPERATION}* indicates that there are outstanding asynchronous operations, but none has yet completed.
- no-outstanding-operation *{MS_NO_OUTSTANDING_OPERATION}* indicates that there are no outstanding asynchronous operations.

This result is valid if **Status** has the value **success**.

On successful return with **Completion-Flag** having the value **completed-operation**, the Status and the Invoke-ID of the completed operation are returned.

Operation-Status (Status)
Indicates whether the asynchronous operation succeeded or not; if not, the possible error values are those listed for the individual operation in the corresponding function description.

This result is valid if **Status** has the value success and **Completion-Flag** has the value **completed-operation**.

Result (Object(*))
Gives the result of the completed asynchronous operation. Its value will be the constant **Null-Result** *{MS_NULL_RESULT}* if the operation was one that does not return a result. Otherwise, the OM object's OM class is that of the result of the asynchronous operation and can be determined using the OM functions.

Note – The possible forms of "result_return" that `ms_receive_result` is required to support is restricted to the results of interface functions within the FUs supported.

This result is valid if **Status** has the value success and **Completion-Flag** has the value **completed-operation**.

Invoke-ID (Integer)
Specifies the Invoke-ID of the completed asynchronous operation whose result is being returned. This result is valid if **Status** has the value success and **Completion-Flag** has the value **completed-operation**.

Errors

This function can return a **System-Error** or one of the following **Library-Errors**: bad-session, feature-unavailable, feature-not-negotiated, no-workspace, out-of-memory, miscellaneous.

This function does not report any Message Store errors or a **Communications-Error** in its **Status** result. (Any such errors related to the completed asynchronous operation are reported in **Operation-Status**, as described above.)

4.6.9 ms_register

Modifies various parameters held by the MTS regarding delivery of messages to the MS.

Synopsis

```
#include <xom.h>
#include <xms.h>
#include <xmsga.h>

MS_status ms_register (
    OM_private_object  session,
    OM_object          register_argument,
    MS_invoke_id       *invoke_id_return
);
```

Description

This function is used to make long-term modifications to various parameters held by the MTS regarding the delivery of messages to the MS.

Arguments

Session (Object(Session))
Specifies the MS session over which this function is performed.

Register-Argument (Object(Register-Argument))
Specifies how to modify (using the `ms_register` function) various parameters held by the MTS regarding delivery of messages to the MS. These parameters include the following:

- User-Name
  specifies the OR-name of the MS (this corresponds to the OR-name of the UA), if the user-name is to be changed.
- User-Address
  specifies the OR-address of the MS (this corresponds to the OR-name of the UA), if it is required by the MTS and if it is to be changed.
- Deliverable-Encoded-Information-Types
  indicates the encoded-information-types that the MTS shall permit to appear in messages delivered to the MS, if they are to be changed. The MTS shall reject as undeliverable any message for an MS for which the MS

is not registered to accept delivery of all of the encoded-information-types of the message. Note that the MS may register to receive the undefined encoded-information-type. This argument, Deliverable-Encoded-Information-Types, also indicates the possible encoded-information-types to which implicit conversion can be performed.

- Deliverable-Maximum-Content-Length
  indicates the content-length, in octets, of the longest content message that the MTS shall permit to appear in messages being delivered to the MS, if it is to be changed. The MTS shall reject as undeliverable any message for an MS for which the MS is not registered to accept delivery of messages of its size.
- Default-Delivery-Controls
  indicates default delivery controls which are registered using the `ms_register` function. The default delivery control arguments shall not admit messages whose delivery are prohibited by the prevailing registered values of the Deliverable-Encoded-Information-Types argument, the Deliverable-Content-Types argument or the Deliverable-Maximum-Content-Length argument.
- Deliverable-Content-Types
  indicates the content-types that the MTS shall permit to appear in messages delivered to the MS, if they are to be changed. The MTS shall reject as undeliverable any message for an MS for which the MS is not registered to accept delivery of all of the content-types of the message. Note that the MS may register to receive the undefined content-type.
- Labels-And-Redirections contains one or both of the following:

  the OR-name of an alternate recipient to which messages are to be redirected, if this is to be changed.

  the security-labels of the UA, if they are to be changed.

Results

Status (Status)
Whether the function succeeded or not, if used synchronously; or whether the function has been initiated, if used asynchronously.

Invoke-ID (Integer)
The Invoke-ID of the asynchronous operation.

Errors

This function can return a **System-Error** or one of the following **Library-Errors**: bad-argument, bad-session, asynchrony-not-supported, feature-unavailable, feature-not-negotiated, no-workspace, out-of-memory, miscellaneous, too-many-operations.

The following Message Store errors may be returned: register-rejected.

This function can return a **Communications-Error**.

4.6.10 `ms_register_ms`

Registers or deregisters various information with the MS.

Synopsis

```
#include <xom.h>
#include <xms.h>
#include <xmsga.h>

MS_status ms_register_ms(
    OM_private_object   session,
    OM_object           register_ms_argument,
    MS_invoke_id       *invoke_id_return
);
```

Description

This function is used to register or deregister various information with the MS.

Arguments

Session (Object(Session))
Specifies the MS session over which this function is performed.

Register-MS-Argument (Object(Register-MS-Argument))
Specifies the argument for the `ms_register_ms` function. It comprises:

- Auto-Action-Registrations
  specifies a set of auto-action-registrations, one for each auto-action to be registered.
- Auto-Action-Deregistrations
  specifies a set of auto-action-deregistrations, one for each auto-action to be deregistered.
- List-Attribute-Defaults
  specifies a default set of attribute-types to indicate which attributes should be returned for any subsequent `ms_list` function if the entry-information-selection argument is absent. This value replaces any previously registered default set. If absent, no change will be applied to the registered default set.

- Fetch-Attribute-Defaults
  specifies a default set of attribute-types to indicate which attributes should be returned for any subsequent `ms_fetch` function if the entry-information-selection argument is absent. This value replaces any previously registered default set. If absent, no change will be applied to the registered default set.
- Change-Credentials
  specifies the old and new credentials of the end user, if change credentials was requested.
- User-Security-Labels
  contains the security-labels of the UA, if they are to be changed.

Results

Status (Status)
Indicates whether the function succeeded or not, if used synchronously; or whether the function has been initiated, if used asynchronously.

Invoke-ID (Integer)
Specifies the Invoke-ID of the asynchronous operation.

Errors

This function can return a **System-Error** or one of the following **Library-Errors**: bad-argument, bad-session, bad-class, no-such-class, asynchrony-not-supported, feature-unavailable, feature-not-negotiated, out-of-memory, miscellaneous, too-many-operations.

The following Message Store errors may be returned: attribute-error, auto-action-request-error, invalid-parameter-error, no-workspace, security-error, service-error.

This function can return a **Communications-Error**.

4.6.11 ms_shutdown

Shuts down the interface, discarding the workspace.

Synopsis

```
#include <xom.h>
#include <xms.h>
#include <xmsga.h>

void ms_shutdown (
    void
);
```

Description

This function shuts down the interface established by `ms_initialize`, and may enable the service to release resources (in its workspace). All sessions will become invalid and no other MS interface functions should be called after this function except for `ms_initialize`. This function does not return any results or errors.

Arguments

None.

Results

None.

Errors

None.

4.6.12 ms_submit

Submits a communique (message or probe).

Synopsis

```
#include <xom.h>
#include <xms.h>
#include <xmsga.h>

MS_status ms_submit (
    OM_private_object   session,
    OM_object           communique,
    OM_private_object  *submission_results_return,
    MS_invoke_id       *invoke_id_return
);
```

Description

After verifying the integrity of the communique, this function submits a communique (message or probe) by adding it to the submission queue to which the current session provides access. A message may be submitted by requesting the forwarding of a delivered message identified by its sequence-number.

Arguments

Session (Object(Session))
Specifies the MS session over which this function is performed.

Communique (Object(Submitted-Communique))
Specifies the object (a message or a probe) to be submitted. Its purported originator shall be the user associated with the session. If the communique is a private object, it is made inaccessible to the client; and is deleted at the discretion of the service.

Note – If the subclass, **Item-To-Forward**, were used for this argument, it is possible to request forwarding a delivered message identified by its MS sequence-number. (The MS entry to be thus forwarded should be a delivered message entry. Forwarding of entries that are not delivered messages is not defined in this specification.)

Results

Status (Status)
Indicates whether the function succeeded or not, if used synchronously; or whether the function has been initiated, if used asynchronously.

Result (Object(Submission-Results))
On successful completion of a synchronous call, contains the results of the submission. For details, see Chapter 2, “Message Access (MA) Interface”..

Invoke-ID (Integer)
Specifies the Invoke-ID of the asynchronous operation.

Errors

This function can return a **System-Error** or one of the following **Library-Errors**: bad-argument, bad-session, bad-class, no-such-class, asynchrony-not-supported, feature-unavailable, feature-not-negotiated, out-of-memory, miscellaneous, too-many-operations.

The following Message Store errors may be returned: sequence-number-error, submission-control-violated, element-of-service-not-subscribed, no-workspace, originator-invalid, recipient-improperly-specified, inconsistent-request, security-error, unsupported-critical-function, remote-bind-error.

This function can return a **Communications-Error**.

4.6.13 ms_summarize

Summarizes counts of selected entries in an information base.

Synopsis

```
#include <xom.h>
#include <xms.h>
#include <xmsga.h>

MS_status ms_summarize (
    OM_private_object   session,
    OM_object           summarize_argument,
    OM_private_object  *summarize_result_return,
    MS_invoke_id       *invoke_id_return
);
```

Description

This function is used to obtain summary counts of selected entries in an information base; in addition, a count of entries selected and their lowest and highest sequence-numbers are also returned. Zero or more individual summaries may be requested.

Arguments

Session (Object(Session))
Specifies the MS session over which this function is performed.

Summarize-Argument (Object(Summarize-Argument))
Specifies the argument for the `ms_summarize` function. It comprises:

- Information-Base-Type
  specifies the repository containing entries for delivered messages and reports. Its value is stored-messages *{MS_STORED_MESSAGES}*
- Selector
  specifies the set of criteria for determining the entries which are to be summarised.
- Summary-Requests
  indicates the sequence of Attribute-Types for which summaries are requested.

Results

Status (Status)
Indicates whether the function succeeded or not, if used synchronously; or whether the function has been initiated, if used asynchronously.

Result (Object(Summary-Result))
On successful completion of a synchronous call, contains one or more of the following:

Next
is returned in the case where the number of entries selected would have been greater if it were not for the limit specified in the selector. This would then give the sequence-number for the next entry that would have been selected.

Count
gives the number of entries that matched the selection criteria.

Span
gives the range of sequence-numbers of entries that matched the selection criteria. It is absent if there were no such entries (Count value is zero).

Summaries
is a sequential list of summaries; one for each summary-request. The summaries are returned in the order that they were requested in the Summary function.

Invoke-ID (Integer)
Specifies the Invoke-ID of the asynchronous operation.

Errors

This function can return a **System-Error** or one of the following **Library-Errors**: bad-argument, bad-session, bad-class, no-such-class, asynchrony-not-supported, feature-unavailable, feature-not-negotiated, out-of-memory, miscellaneous, too-many-operations.

The following Message Store errors may be returned: attribute-error, invalid-parameter-error, no-workspace, range-error, security-error, sequence-number-error, service-error.

This function can return a **Communications-Error**.

4.6.14 `ms_unbind`

Terminates a session with the Message Store.

Synopsis

```
#include <xom.h>
#include <xms.h>
#include <xmsga.h>

MS_status ms_unbind (
    OM_private_object  session
);
```

Description

This function terminates a session with the Message Store.

This implies that the results of any outstanding asynchronous operations initiated in this session can no longer be received and it is not possible to know whether or not such operations succeeded. Any such operations may have been carried out or terminated prematurely. For this reason, it is recommended that all outstanding asynchronous operations be processed using `ms_receive_result` before calling `ms_unbind`.

Arguments

Session (Object(Session))
Specifies the MS session over which this function is performed.

Results

Status (Status)
Indicates whether the function succeeded or not.

Errors

This function can return a **System-Error** or one of the following **Library-Errors**: bad-session, no-workspace, out-of-memory, miscellaneous.

This function can return a **Communications-Error**.

4.6.15 ms_wait

Returns when a new entry is available in the Message Store for retrieval or when a period of time has elapsed, whichever occurs first.

Synopsis

```
#include <xom.h>
#include <xms.h>
#include <xmsga.h>

MS_status ms_wait (
    OM_private_object   session,
    OM_uint32           interval,
    OM_private_object  *wait_result_return,
    MS_invoke_id       *invoke_id_return
);
```

Description

This function returns when a new entry is available in the Message Store for retrieval or when a period of time has elapsed, whichever occurs first. (see reference X.413, for the definition of *new*, a possible value for the Entry-Status MS attribute.)

Arguments

Session (Object(Session))
Specifies the MS session over which this function is performed.

Interval (Integer)
Specifies the maximum length of time (in milliseconds) that the service is to wait before returning a result when checking for entries in the Message Store available for retrieval.

Results

Status (Status)
Indicates whether the function succeeded or not, if used synchronously; or whether the function has been initiated, if used asynchronously.

Result (Object(Wait-Result))
On successful completion of a synchronous call, will be either:

the constant No-New-Entries *{MS_NO_NEW_ENTRIES}* if there are no new MS entries available for retrieval.

a private object if there are new MS entries available for retrieval.

Invoke-ID (Integer)
Specifies the Invoke-ID of the asynchronous operation.

Errors

This function can return a **System-Error** or one of the following **Library-Errors**: bad-argument, bad-session, asynchrony-not-supported, feature-unavailable, feature-not-negotiated, no-workspace, out-of-memory, miscellaneous, too-many-operations.

This function can return a **Communications-Error**.

4.7 Use And Implementation of the MS Interface

If an argument to a function has an invalid value (such as a value outside the domain of the function or a pointer outside the address space of the program), the behaviour is undefined.

Any function declared in a header may be implemented as a macro defined in the header so a library function should not be declared explicitly if its header is included.

Any macro definition of a function can be suppressed locally by enclosing the name of the function in parentheses because the name is not then followed by the left parenthesis that indicates expansion of a macro function name. For the same syntactic reason, it is permitted to take the address of a library function even if it is also defined as a macro. The use of *#undef* to remove any macro definition will also ensure that an actual function is referred to. Any invocation of a library function that is implemented as a macro will expand to code that evaluates each of its arguments exactly once, fully protected by parentheses where necessary; hence it is generally safe to use arbitrary expressions as arguments. Likewise, those function-like macros described in the following sections may be invoked in an expression wherever a function with a compatible return data type could be called.

4.7.1 Function Return Values

The return value of a C function is always bound to the **Status** result of the language-independent description. Functions return a value of type *MS_status*, which is an error indication. If, and only if, the function succeeds, its value will be **success**, expressed in C by the constant *{MS_SUCCESS}*. If a function returns with a status other than this, then it has not updated the return parameters. The value of the status, in this case, is an error as described in Chapter 9, “Message Store Package Errors”.

Since C does not provide multiple return values, functions must return all other results by writing into storage passed by the client program. Any argument that is a pointer to such storage has a name ending with *_return*. For example, the C parameter declaration *Uint *completion_flag_return* indicates that the function will return an unsigned integer **Completion-Flag** as a result, so the actual argument to the function must be the address of a suitable variable. This notation allows the reader to distinguish between an input parameter that happens to be a pointer and an output parameter where the '*' is used to simulate the semantics of passing by reference.

4.7.2 Sequence Of MS Interface Functions

The MS interface has an initialization and shutdown sequence that allows the negotiation of optional features. This involves the functions `ms_initialize` and `ms_shutdown`.

Every client program must first call `ms_initialize` which returns a workspace. By default, this workspace supports the standard MS Package (see Chapter 8, "Message Store Package" and Chapter 9, "Message Store Package Errors") and the MS General Attributes Package (see Chapter 10, "Message Store General Attributes Package"). The workspace can be extended to support the optional MS Interpersonal Messaging Attributes Package (see Chapter 11, "Message Store IM Attributes Package") or any vendor extensions. All such packages or other extensions are identified by means of OSI Object Identifiers.

Once the interface has been initialized, `ms_initialize` may not be called again until `ms_shutdown` has been called. Figure 4-1 shows the order and context in which each MS interface function can be used.

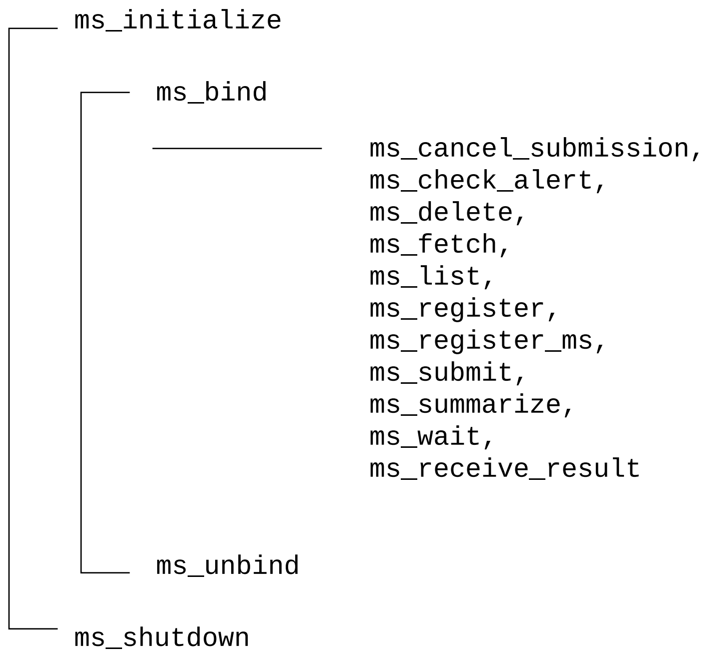

Figure 4-1 Sequence of MS Interface Functions

After negotiating a workspace with the required features, the client can use the workspace as required. It can create and manipulate OM objects using the OM functions and can start one or more MS sessions using `ms_bind`.

Each session is terminated using `ms_unbind`, and all its OM objects released using `OM-delete`; finally the client should ensure that resources associated with the interface are released by calling `ms_shutdown`.

It is possible to retain access to service-generated public objects after `ms_shutdown` has been called or to start another cycle by calling `ms_initialize` if desired.

Message Handling Packages 5

This chapter defines the MH 84 Package and the MH 88 Package. The entire chapter applies to both packages, except where indicated.

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.

5.1 MH Class Hierarchy

This section depicts the hierarchical organization 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 (see the *Solstice XOM Programming Reference Manual*)

- Algorithm
  - — Algorithm and Result
- Attribute (Class is defined in Chapter 8, "Message Store Package")
  - — AVA (Class is defined in Chapter 8, "Message Store Package")
  - — Filter-Item (Class is defined in Chapter 8, "Message Store Package")
- *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 Delivery Report [A]
    - Delivered Per-recipient Non-Delivery Report [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
- *Name*
    - DS-DN
- **OR Address**
    - **OR Name**
- Redirection Record
- *Relative-Name*
    - DS-RDN
- Security Label
- Session
- Teletex NBPs
- *Token Public Data*
    - MT Public Data

5.2 MH Class Definitions

This section defines the MH classes. It describes the attributes specific to a class in a table like those used in the *Solstice XOM Programming Reference Manual*. 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.*

5.2.1 Algorithm

An instance of class **Algorithm** identifies a mathematical (typically cryptographic) algorithm. This class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 5-1.

Table 5-1 Attributes Specific to Algorithm

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

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.

5.2.2 Algorithm and Result

An instance of class **Algorithm and Result** identifies a mathematical (typically cryptographic) algorithm and provides its result on a particular occasion. This class has the OM attributes of its superclasses *Object*, **Algorithm** and additionally, the OM attributes listed in Table 5-2.

Table 5-2 Attributes Specific to Algorithm and Result

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

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

5.2.3 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. This class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 5-3.

Table 5-3 Attributes Specific to Asymmetric Token

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

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

5.2.4 Attribute

As defined in Section 8.2.3, "Attribute," on page 8-6.

5.2.5 AVA

As defined in Section 8.2.6, "AVA," on page 8-8.

5.2.6 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*.

This class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 5-4.

Table 5-4 Attributes Specific to Bilateral Information

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.

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.

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.

5.2.7 Communique

An instance of class **Communique** is a primary information object conveyed by users via the MTS. This class has the OM attributes of its superclasses *Object*, **Extensible Object** and additionally, the OM attributes listed in Table 5-5.

Table 5-5 Attributes Specific to Communique

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 Id)[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)	-	1-512	-	-
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 *XDS Specification* (see Appendix C, "Referenced Documents").

Alternate Recipient Allowed
Whether the originator permits the MTS to deliver the subject message to an alternate recipient. A management domain (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 recognized 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**—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 that the value **unidentified** was introduced to X.400 (1984) by means of the *X.400-Series Implementors' Guide*.

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.

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.

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 **reassigned**.

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

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.

5.2.8 Content

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

The purpose of this abstract class is to provide a common superclass for General Content (defined in Section 5.2.26, “General Content,” on page 5-36), Interpersonal Message and Interpersonal Notification (defined in Section 6.2, “IM Class Definitions”), 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.

This class has the OM attributes of its superclass *Object* and no additional OM attributes.

5.2.9 Delivered Message

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

This class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 5-6.

Table 5-6 Attributes Specific to Delivered Message

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

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.

5.2.10 Delivered Per-recipient Delivery Report

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.

This class has the OM attributes of its superclasses *Object*, **Extensible Object**, **Delivered Per-recipient Report** and additionally, the OM attributes listed in Table 5-7.

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

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 *XDS* Specification (see Appendix C, “Referenced Documents”).

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 5.3.4, “Delivery Point,” on page 5-84.

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.

5.2.11 Delivered Per-recipient Non-Delivery Report

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.

This class has the OM attributes of its superclasses *Object*, **Extensible Object**, **Delivered Per-recipient Report** and additionally, the OM attributes listed in Table 5-8.

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

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

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 5.3.5, "Diagnostic," on page 5-85.

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 5.3.10, "Reason," on page 5-91.

5.2.12 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.

This class has the OM attributes of its superclasses *Object*, **Extensible Object** and additionally, the OM attributes listed in Table 5-9.

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

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.

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 characterize, or would characterize, 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.

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.

5.2.13 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.

This class has the OM attributes of its superclasses *Object*, **Extensible Object, Delivery Report** and additionally, the OM attributes listed in Table 5-10.

Table 5-10 Attributes Specific to Delivered Report

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

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

5.2.14 Delivery Confirmation

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

This class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 5-11.

Table 5-11 Attributes Specific to Delivery Confirmation

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 *XDS Specification* (see Appendix C, "Referenced Documents").

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.

5.2.15 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.

This class has the OM attributes of its superclasses *Object*, **Extensible Object** and additionally, the OM attributes listed in Table 5-12.

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[5]	-	-
Bureau FAX Delivery	Boolean	-	0-1[5]	false	1988
Confidentiality Algorithm	Object (Algorithm)	-	0-1	-	1988

Table 5-12 Attributes Specific to Delivery Envelope

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
Content Identifier	String (Printable)	1-16	0-1	-	1988
Content Type	Integer or String (Object Identifier)[1]	-	0-1[5]	P2-1984	-
Conversion Loss Prohibited	Boolean	-	0-1[5]	false	1988
Conversion Prohibited	Boolean	-	0-1[5]	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	-	-
Originally Intended Recip	Object (OR Name)	-	0-1	-	-
Originator Certificate	Object (Certificates)[3]	-	0-1	-	1988
Originator Name	Object (OR Name)	-	0-1[5]	-	-
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)	-	1-10	any	1988
Priority	Enum (Priority)	-	0-1[5]	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[5]	-	-

Table 5-12 Attributes Specific to Delivery Envelope

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
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 *XDS Specification* (see Appendix C, "Referenced Documents").

[4] This attribute shall be absent only when an object of this class is a sub-object of a Message Body Part object (see "Message Body Part" on page 6-14).

[5] This attribute may be absent only when an object of this class is a sub-object of a Message Body Part object (see "Message Body Part" on page 6-14).

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.

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
Correlates the message with the delivery report that it produces. 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 content type 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 characterize 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 that this attribute has no counterpart in X.400. Its purpose is to support the `ma_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 Recipient
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 5.3.7, "Postal Mode," on page 5-89.

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 5.3.8, "Postal Report," on page 5-90.

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 5.3.9, "Priority," on page 5-90.

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 5.3.12, "Registration," on page 5-92.

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.

5.2.16 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).

This class has the OM attributes of its superclasses *Object*, **Extensible Object** and additionally, the OM attributes listed in Table 5-13.

Table 5-13 Attributes Specific to Delivery Report

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)	-	0-512	-	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 *XDS Specification* (see Appendix C, "Referenced Documents").

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.

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. It identifies the content of that communique.

Content Type
Identifies the syntax and semantics of the Content attribute of the submitted 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 that this attribute has no counterpart in X.400. Its purpose is to support the `ma_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 shall be present if the like-named attribute 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 of the delivery report.

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 of the submitted communique.

Subject MTS Identifier
The MTS identifier of the submitted communique.

5.2.17 DS-DN

An instance of OM class **DS-DN** represents a name of a directory object.

This OM class has the OM attributes of its superclasses (*Object*, *Name*) and additionally the OM attributes listed in the table in Table 5-14 below.

Table 5-14 OM Attributes of a DS-DN

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
RDNs	Object(DS-RDN)	-	0 or more	-

RDNs

The sequence of RDNs that define the path through the DIT from its root to the object that the DS-DN denotes. The DS-DN of the root of the directory is the null name (no RDNs values). The order of the values is significant; the first value is closest to the root, and the last value is the RDN of the object.

This class belongs to the Directory Services package, defined in the XDS Specification.

5.2.18 DS-RDN

An instance of OM class **DS-RDN** is a relative distinguished name (RDN). An RDN uniquely identifies an immediate subordinate of an object whose entry appears in the DIT.

This OM class has the OM attributes of its superclasses (*Object*, *Relative-Name*) and additionally the OM attributes listed in Table 5-15 below.

Table 5-15 OM Attributes of an RDN

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
AVAs	Object(AVA)	-	1 or more	-

AVAs

The AVAs that are marked by the DIB as components of the object's RDN. The assertions shall be true of the object but of none of its siblings, and the attribute types and values they contain shall appear in the object's directory entry. The order of the AVAs is not significant.

This class belongs to the Directory Services package, defined in the XDS Specification.

5.2.19 EITs

An instance of class **EITs** summarizes 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.

This class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 5-16.

Table 5-16 Attributes Specific to EITs

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
Builtin EITs	Enum (Builtin EIT)	-	0-9	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.

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 5.3.2, “Builtin EIT,” on page 5-82.

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 Table 5-17, 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.

Table 5-17 Selected Values of the External EITs Attribute

Value	Object Identifier (ASN.1)
g3-fax	{joint-iso-ccitt mhs-motis(6) mts(3) 4 3}
g4-class-1	{joint-iso-ccitt mhs-motis(6) mts(3) 4 4}
ia5-text	{joint-iso-ccitt mhs-motis(6) mts(3) 4 2}
mixed-mode	{joint-iso-ccitt mhs-motis(6) mts(3) 4 9}
teletex	{joint-iso-ccitt mhs-motis(6) mts(3) 4 5}
telex	{joint-iso-ccitt mhs-motis(6) mts(3) 4 1}
undefined	{joint-iso-ccitt mhs-motis(6) mts(3) 4 0}
videotex	{joint-iso-ccitt mhs-motis(6) mts(3) 4 6}

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 characterizes 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.

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 characterize 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.

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 characterize the teletex documents. This attribute is present only if teletex is among the values of either the Builtin EITs or the External EITs attribute.

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.

5.2.20 Expansion Record

An instance of class **Expansion Record** documents the expansion of a DL or, occasionally, the submission of a communique.

This class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 5-18.

Table 5-18 Attributes Specific to Expansion Record

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
Name	Object (OR Name)	-	1	-	1988
Time	String (UTC Time)	0-17	1	-	1988

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.

5.2.21 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.

This class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 5-19.

Table 5-19 Attributes Specific to Extensible Object

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

Table 5-19 Attributes Specific to Extensible Object

Attribute	Value Syntax	Value Length	Value Number	Value Initially	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.

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.

5.2.22 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.

This class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 5-20.

Table 5-20 Attributes Specific to Extension

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.

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
Parameterizes the extended processing.

5.2.23 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.

This class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 5-21.

Table 5-21 Attributes Specific to External Trace Entry

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.

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.

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.

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 5.3.1, “Action,” on page 5-82. 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 characterize, or would characterize, 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.

5.2.24 Filter Item

As defined in Section 8.2.31, “Filter-Item,” on page 8-28.

5.2.25 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.

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.

This class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 5-22.

Table 5-22 Attributes Specific to G3 Fax NBPs

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	-

All Attributes
Whether the like-named G3 facsimile capability is required for the images.

5.2.26 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.

This class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 5-23.

Table 5-23 Attributes Specific to General Content

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
Binary Content	String (octet)	-	1	-	-

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.

5.2.27 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.

This class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 5-24.

Table 5-24 Attributes Specific to Internal Trace Entry

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.

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.

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 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 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 MTA name is present.

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 5.3.1, “Action,” on page 5-82. 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 characterize, or would characterize, 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.

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 recognized by various functional standards, or profiles, based upon X.400 (1984). The scope of such recognition is outside the scope of this document.

5.2.28 Local Delivery Confirmation

An instance of class **Local Delivery Confirmation** is an acknowledgment of the delivery of a message (but not a report) to the UA of a local user.

This class has the OM attributes of its superclasses *Object*, **Delivery Confirmation** and additionally, the OM attributes listed in Table 5-25.

Table 5-25 Attributes Specific to Local Delivery Confirmation

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
Recipient Number	Integer	-	1	0	1988

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.

5.2.29 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.

This class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 5-26.

Table 5-26 Attributes Specific to Local Delivery Confirmations

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
Delivery Confirmations	Object (Local Delivery Confirm)	-	0-32767	-	1988

Delivery Confirmations
A local delivery confirmation for each local recipient to which the local DR applies.

5.2.30 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.

This class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 5-27.

Table 5-27 Attributes Specific to Local NDR

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
Per-recipient Reports	Object (Local Per-recipient NDR)	-	0-32767	-	-

Per-recipient Reports
A local per-recipient NDR for each local recipient to which the local NDR applies.

5.2.31 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.

This class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 5-28.

Table 5-28 Attributes Specific to Local Per-recipient NDR

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	-
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.

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 5.3.5, “Diagnostic,” on page 5-85. 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 5.3.10, “Reason,” on page 5-91. 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.

Temporary
Whether the factor that prevented the subject message from being conveyed to the recipient is temporary, rather than permanent.

5.2.32 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.

This class has the OM attributes of its superclasses *Object*, **Extensible Object, Communique** and additionally, the OM attributes listed in Table 5-29.

Table 5-29 Attributes Specific to Message

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	-

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.

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 Return Requested
Whether the Content attribute is to be included as the content return requested 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 5.3.9, "Priority," on page 5-90.

5.2.33 Message RD

An instance of class **Message RD** identifies an intended recipient of a message and records certain information about that recipient

This class has the OM attributes of its superclasses *Object*, **Extensible Object**, **Submitted Probe RD**, **Submitted Message RD** and additionally, the OM attributes listed in Table 5-30.

Table 5-30 Attributes Specific to Message RD

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**.

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 5.3.13, "Report Request," on page 5-92. 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. 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.

5.2.34 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.

This class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 5-31.

Table 5-31 Attributes Specific to MT Public Data

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

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.

5.2.35 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.

This class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 5-32.

Table 5-32 Attributes Specific to MTS Identifier

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.

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. The defined values of this attribute may be identical to those of the PRMD Name attribute of an O/R name of a user served by the originating domain.

5.2.36 Name

The OM class **Name** represents a name of an object in the directory, or a part of such a name.

It is an abstract class, which has the attributes of its superclass *Object* and no other OM attributes.

A name unambiguously distinguishes the object from all other objects whose entries appear in the DIT. However, an object may have more than one name, that is, a name need not be unique. A distinguished name is unique; there are no other distinguished names that identify the same object. A relative distinguished name is a part of a name, and only distinguishes the object from others that are its siblings. Most of the interface functions take a name argument, the value of which must be an instance of one of the subclasses of this OM class. Thus, this OM class serves to collect together all possible representations of names.

This book defines one subclass of this OM class, and thus a single representation for names:

- **DS-DN**, which provides a representation for names, including distinguished names

This class belongs to the Directory Services package, defined in the XDS Specification.

5.2.37 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.

This class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 5-33.

Table 5-33 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	-	-
Organization Name	String (Printable or Teletex^{3,4})	1-64	0-2$^{5, 7}$	-	-
Organizational Unit Name 1	String (Printable or Teletex^{3,4})	1-32	0-2^{5}	-	-
Organizational Unit Name 2	String (Printable or Teletex^{3,4})	1-32	0-2^{5}	-	-
Organizational Unit Name 3	String (Printable or Teletex^{3,4})	1-32	0-2^{5}	-	-
Organizational 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

Table 5-33 Attributes Specific to OR Address

Attribute	Value Syntax	Value Length	Value Number	Value Initially	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 Organization 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 *XDS Specification* (see Appendix C, "Referenced Documents").

[3] If only one value is present in international communication, its syntax shall be String (Printable). 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. Printable strings are required internationally to ensure that such device limitations do not prevent communication.

[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 Organizational Unit 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.

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, organization 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, without punctuation (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.

Organization Name
The name of the user’s or DL’s organization. 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 organizations together.

Organizational Unit Name 1
The name of a unit (for example, division or department) of the organization that the Organization Name attribute denotes. The attribute’s values are defined by that organization.

Organizational Unit Name 2
The name of a subunit (for example, division or department) of the unit that the Organizational Unit Name 2 attribute denotes. The attribute's values are defined by the latter unit.

Organizational Unit Name 3
The name of a subunit (for example, division or department) of the unit that the Organizational Unit Name 2 attribute denotes. The attribute's values are defined by the latter unit.

Organizational Unit Name 4
The name of a subunit (for example, division or department) of the unit that the Organizational 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 Organization Name
The name of the organization through which the user takes delivery of physical messages.

Postal Patron Details
Additional information (for example, the name of the organizational 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 5.3.15, Terminal Type.

X121 Address
The network address of the user’s terminal. Its values are defined by X.121.

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 Organizational Unit Name 1, Organizational Unit Name 2, Organizational Unit Name 3 and Organizational Unit Name 4 attributes constitute the “organizational unit name attributes.” The ISDN Number, ISDN Subaddress, Presentation Address and X.121 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).

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 summarized in Table 5-34. 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.

Table 5-34 Forms of O/R Address

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	-	-	-
organization 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	-

Table 5-34 Forms of O/R Address

Attribute	**Mnem**[1]	**Num**[2]	**Spost**[3]	**Upost**[4]	**Term**[5]
Postal General Delivery Address	-	-	C	-	-
Postal Locale	-	-	C	-	-
Postal Office Box Number	-	-	C	-	-
Postal Office Name	-	-	C	-	-
Postal Office Number	-	-	C	-	-
Postal organization 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).

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 organization Name attribute, the Organizational 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, organization 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. There are two kinds of postal O/R address:

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.

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.

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.

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.

5.2.38 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.

This class has the OM attributes of its superclasses *Object*, **OR Address** and additionally, the OM attributes listed in Table 5-35.

Table 5-35 Attributes Specific to OR Name

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
Directory Name	Object (Name)[1]	-	0-1	-	1988

[1]As defined in Section 5.2.36, "Name," on page 5-47.

Directory Name
The name assigned to the user or DL by the world-wide (in other words, X.500) directory.

5.2.39 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.

This class has the OM attributes of its superclasses *Object*, **Extensible Object**, **Per-recipient Report** and additionally, the OM attributes listed in Table 5-36.

Table 5-36 Attributes Specific to Per-recipient DR

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 *XDS Specification* (see Appendix C, "Referenced Documents").

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 5.3.4, "Delivery Point," on page 5-84.

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.

5.2.40 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.

This class has the OM attributes of its superclasses *Object*, **Extensible Object**, **Per-recipient Report** and additionally, the OM attributes listed in Table 5-37.

Table 5-37 Attributes Specific to Per-recipient NDR

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	-

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 "Diagnostic" on page 5-85.

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 5.3.10, "Reason," on page 5-91.

5.2.41 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.

This class has the OM attributes of its superclasses *Object*, **Extensible Object** and additionally, the OM attributes listed in Table 5-38.

Table 5-38 Attributes Specific to Per-recipient Report

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.

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 characterize, or would characterize, 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 Recipient
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 5.3.13, "Report Request," on page 5-92.

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. 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.

5.2.42 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.

This class has the OM attributes of its superclasses *Object*, **Extensible Object**, **Communique** and additionally, the OM attributes listed in Table 5-39.

Table 5-39 Attributes Specific to Probe

Attribute	**Value Syntax**	**Value Length**	**Value Number**	**Value Initially**	**1988?**
Content Length	Integer	-	0-1	-	-

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.

5.2.43 Probe RD

An instance of class **Probe RD** identifies an intended recipient of a probe (or message) and records certain information about that recipient.

This class has the OM attributes of its superclasses *Object*, **Extensible Object**, **Submited Probe RD** and additionally, the OM attributes listed in Table 5-40.

Table 5-40 Attributes Specific to Probe RD

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**.

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 5.3.13, “Report Request,” on page 5-92.

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. 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.

5.2.44 RD

An instance of class **RD** identifies an intended recipient of a communique and records certain information about that recipient.

This class has the OM attributes of its superclasses *Object*, **Extensible Object**, and no additional OM attributes.

5.2.45 Redirection Record

An instance of class **Redirection Record** documents a communique's redirection.

This class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 5-41.

Table 5-41 Attributes Specific to Redirection Record

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

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 5.3.11, “Redirection Reason,” on page 5-91.

Time
The date and time at which redirection occurred.

5.2.46 Relative-Name

The OM class *Relative-Name* represents the Relative Distinguished Names of objects in the directory.

It is an abstract class, which has the attributes of its superclass (*Object*) and no other OM attributes.

A relative distinguished name (RDN) is a part of a name, and only distinguishes the object from others that are its siblings.

This OM class serves to collect together all possible representations of RDNs. An argument of interface functions that is an RDN, or an OM attribute value that is an RDN, will be an instance of one of the subclasses of this OM class.

There is one subclass of this OM class, and thus a single representation for RDNs:

- **DS-RDN**, which provides a representation for relative distinguished names

5.2.47 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).

This class has the OM attributes of its superclasses *Object*, **Extensible Object**, **Delivery Report** and additionally, the OM attributes listed in Table 5-42.

Table 5-42 Attributes Specific to Report

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
External Trace Info	Object (External Trace Entry)	-	1-512	-	-
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	-	-

Table 5-42 Attributes Specific to Report

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
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.

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.

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.

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.

Report Destination
The O/R name of the report's immediate destination. The MTA that originates the report initializes 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.

5.2.48 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.

This class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 5-43.

Table 5-43 Attributes Specific to Security Label

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

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 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 5.3.14, “Security Classification,” on page 5-93.

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.

5.2.49 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.”

This class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 5-44.

Table 5-44 Attributes Specific to Session

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.

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.

5.2.50 Submission Results

An instance of class **Submission Results** is an acknowledgment of the successful submission to the MTS of a submitted communique. The MTS provides the acknowledgment.

This class has the OM attributes of its superclasses *Object*, **Extensible Object** and additionally, the OM attributes listed in Table 5-45.

Table 5-45 Attributes Specific to Submission Results

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	-	-

Table 5-45 Attributes Specific to Submission Results

Attribute	Value Syntax	Value Length	Value Number	Value Initially	1988?
Proof of Submission	Object (Algorithm and Result)	-	0-1	-	1988
Submission Time	String (UTC Time)	0-17	1	-	-

[1] As defined in the *XDS Specification* (see Appendix C, "Referenced Documents").

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.

5.2.51 Submitted Communique

An instance of class **Submitted Communique** is a primary information object submitted by users to the MTS.

This class has the OM attributes of its superclasses *Object*, **Extensible Object** and additionally, the OM attributes listed in Table 5-46.

Table 5-46 Attributes Specific to Submitted Communique

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 *XDS Specification* (see Appendix C, "Referenced Documents").

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 recognized 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.

5.2.52 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.

This class has the OM attributes of its superclasses *Object*, **Extensible Object**, **Submitted Communique** and additionally, the OM attributes listed in Table 5-47.

Table 5-47 Attributes Specific to Submitted Message

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

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 5.3.9, "Priority," on page 5-90.

Proof of Submission Requested
Whether the originator of the submitted message requires proof of its submission.

5.2.53 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.

This class has the OM attributes of its superclasses *Object*, **Extensible Object**, **Submitted Probe RD** and additionally, the OM attributes listed in Table 5-48.

Table 5-48 Attributes Specific to Submitted Message RD

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

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.

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 5.3.7, "Postal Mode," on page 5-89.

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 5.3.8, "Postal Report," on page 5-90.

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 5.3.12, "Registration," on page 5-92.

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.

5.2.54 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.

This class has the OM attributes of its superclasses *Object*, **Extensible Object**, **Submitted Communique** and additionally, the OM attributes listed in Table 5-49.

Table 5-49 Attributes Specific to Submitted Probe

Attribute	**Value Syntax**	**Value Length**	**Value Number**	**Value Initially**	**1988?**
Content Length	Integer	-	0-1	-	-

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.

5.2.55 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.

This class has the OM attributes of its superclasses *Object*, **Extensible Object** and additionally, the OM attributes listed in Table 5-50.

Table 5-50 Attributes Specific to Submitted Probe RD

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**.

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 5.3.6, "Explicit Conversion," on page 5-89.

Originator Report Request
The kinds of report that the originator requests in the circumstances prescribed for such reports. For its defined values, see Section 5.3.13, "Report Request," on page 5-92.

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 5.3.3, "Delivery Mode," on page 5-83.

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.

5.2.56 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. In the Teletex service, NBPs are conveyed between terminals by means of the Command Document Start (CDS) data structure of T.61.

This class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 5-51.

Table 5-51 Attributes Specific to Teletex NBPs

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.

Control Character Sets
The information that signals, in a CDS, the control character sets non-basic Teletex terminal capability. This attribute is present at the client’s option.

Graphic Character Sets

Miscellaneous Capabilities

Page Formats

Private Use
See Control Character Sets.

5.2.57 Token

An instance of class **Token** gives to the recipient of the token protected security-relevant information.

This class has the OM attributes of its superclass *Object* and no additional OM attributes.

5.2.58 Token Public Data

This is an abstract class, providing for future expansion of other token types.

This class has the OM attributes of its superclass *Object* and no additional OM attributes.

5.3 MH Syntax Definitions

This section defines the MH enumeration syntaxes, i.e., the syntaxes in the Enumeration group that are specific to MH.

5.3.1 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 a communique or report to another MD or MTA, respectively (in the normal way).

rerouted
The MD or MTA attempts but fails to transfer a communique or report to another MD or MTA, respectively (an exceptional condition).

For 1988 its value may be chosen, additionally, from the following:

expanded
The MD or MTA expands a DL.

redirected
The MD or MTA redirects a communique (but not its report).

5.3.2 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 Table 5-52. 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.

Table 5-52 Values of the Enumeration (Builtin EIT) Syntax

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

Table 5-52 Values of the Enumeration (Builtin EIT) Syntax

Value	Meaning	References
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
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.

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.

5.3.3 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.

5.3.4 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 organization 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 (i.e., 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.

5.3.5 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-unrecognized
The recipient's O/R name is unrecognized.

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*, Appendix C, "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 sub-objects, 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-organization-expired
The recipient’s postal O/R name identifies an organization 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 (i.e., 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 (i.e., 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 (i.e., 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 (i.e., 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.

5.3.6 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, telex-to-videotex, videotex-to-ia5-text, videotex-to-teletex, videotex-to-telex.

The value **no-conversion** signifies no conversion. Other values identify conversions by their names.

5.3.7 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 *Recommendation F.170* (see Appendix C, "Referenced Documents").

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).

5.3.8 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.

5.3.9 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**.

5.3.10 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.

5.3.11 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-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.

5.3.12 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.

5.3.13 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.

5.3.14 *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**. The use of these value is dictated by the security policy in force.

5.3.15 *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.

Interpersonal Messaging Packages 6

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.

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. 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.

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.

6.1 IM Class Hierarchy

This section depicts the hierarchical organization 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 *Solstice XOM Programming Reference Manual*)

- *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, “Message Handling Packages”)
  - — Interpersonal Message
  - — *Interpersonal Notification*
    - — Non-receipt Notification
    - — Receipt Notification
- IPM Identifier
- OR Descriptor
- Recipient Specifier

6.2 IM Class Definitions

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

The value length restrictions were introduced to X.400 by means of the *X.400-Series Implementors' Guide*.

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.

6.2.1 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*.

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

This class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 6-1.

Table 6-1 Attributes Specific to Bilaterally Defined Body Part

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

Bilateral Data
The binary data.

6.2.2 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. 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, i.e., change it from one immediate subclass of **Body Part** to another.

This class has the OM attributes of its superclass *Object* and no additional OM attributes.

6.2.3 Content

As defined in the Message Handling Package in Section 5.2.8, "Content," on page 5-12.

6.2.4 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.

This class has the OM attributes of its superclasses *Object* and additionally, the OM attributes listed in Table 6-2.

Table 6-2 Attributes Specific to Externally Defined Body Part

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 *Solstice XOM Programming Reference Manual*.

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 characterizes 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.

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 Appendix C, "Referenced Documents").

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 organizations.

6.2.5 G3 Fax Body Part

An instance of class **G3 Fax Body Part** comprises Group 3 (G3) facsimile images (i.e., pages).

This class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 6-3.

Table 6-3 Attributes Specific to G3 Fax Body Part

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	-	-

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.

6.2.6 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.

This class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 6-4.

Table 6-4 Attributes Specific to G4 Class 1 Body Part

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

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.

6.2.7 General Text Body Part

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

This class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 6-5.

Table 6-5 Attributes Specific to General Text Body Part

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

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.

6.2.8 IA5 Text Body Part

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

This class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 6-6.

Table 6-6 Attributes Specific to IA5 Text Body Part

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

Repertoire
Identifies the character set to which the text is constrained. For its defined values, see Section 6.3.3, “IA5 Repertoire,” on page 6-25.

Text
The text. It may contain lines of any length. Lines may be folded but shall not be truncated for display or printing.

6.2.9 Interpersonal Message

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

An IPM may be likened to a business memo. In fact, X.400 uses the terms “heading” and “body” with reference 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.

This class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 6-7.

Table 6-7 Attributes Specific to Interpersonal Message

Attribute	**Value Syntax**	**Length**	**Number**	**Initial Value**	**1988?**
Authorizing 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.

Authorizing Users

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

If, for example, a manager instructs a secretary to originate an IPM, the secretary, the IPM's originator, might consider the manager the Authorizing 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.

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 Authorizing 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 Authorizing users attach to the IPM. For its defined values, see Section 6.3.4, "Importance," on page 6-25.

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 Authorizing users of the present IPM consider it to make 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. 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 Authorizing 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. 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 Authorizing 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. If this attribute identifies several users, the originator may be included 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 Authorizing 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 Authorizing users consider the IPM to be. For its defined values, see Section 6.3.8, "Sensitivity," on page 6-26.

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. 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. 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.

6.2.10 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.

This class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 6-8.

Table 6-8 Attributes Specific to Interpersonal Notification

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	-	-

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 as such.

6.2.11 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.

This class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 6-9.

Table 6-9 Attributes Specific to IPM Identifier

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.

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.

6.2.12 ISO 6937 Text Body Part

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

This class has the OM attributes of its superclasses *Object*, **Body Part** and additionally, the OM attributes listed in Table 6-10.

Table 6-10 Attributes specific to ISO 6937 Text Body Part

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

Repertoire
Identifies the character set to which the text is constrained. For its defined values, see Section 6.3.5, “ISO 6937 Repertoire,” on page 6-25.

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.

This class is not defined by X.400, but is defined by various functional standards (see *ISO/DIS 9065* in Appendix C, “Referenced Documents”).

6.2.13 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.

This class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 6-11.

Table 6-11 Attributes Specific to Message Body Part

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

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

6.2.14 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.

This class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 6-12.

Table 6-12 Attributes Specific to Mixed-Mode Body Part

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

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.

6.2.15 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*.

This class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 6-13.

Table 6-13 Attributes Specific to Nationally Defined Body Part

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

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

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.

6.2.16 Non-receipt Notification

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

This class has the OM attributes of its superclasses *Object*, **Interpersonal Notification** and additionally, the OM attributes listed in Table 6-14.

Table 6-14 Attributes Specific to Non-receipt Notification

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, a length of zero is discouraged.

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 6.3.2, "Discard Reason," on page 6-24.

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 6.3.6, "Non-receipt Reason," on page 6-26.

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.

6.2.17 Office Document Architecture Body Part

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

This class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 6-15.

Table 6-15 Attributes Specific to ODA Body Part

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	-	-

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.

6.2.18 OR Descriptor

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

This class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 6-16.

Table 6-16 Attributes Specific to OR Descriptor

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, a length of zero is discouraged.

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.

6.2.19 Receipt Notification

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

This class has the OM attributes of its superclasses *Object*, **Interpersonal Notification** and additionally, the OM attributes listed in Table 6-17.

Table 6-17 Attributes Specific to Receipt Notification

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

Acknowledgment Mode
Identifies how the RN was originated. For its defined values, see Section 6.3.1, "Acknowledgment Mode," on page 6-24.

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.

6.2.20 Recipient Specifier

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

This class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 6-18.

Table 6-18 Attributes Specific to Recipient Specifier

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	-

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 6.2.10, “Interpersonal Notification,” on page 6-11). For its defined values, see Section 6.3.7, “Notification Request,” on page 6-26.

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.

6.2.21 Teletex Body Part

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

This class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 6-19.

Table 6-19 Attributes Specific to Teletex Body Part

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	-

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 (i.e., 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.

6.2.22 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.

This class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 6-20.

Table 6-20 Attributes Specific to Unidentified Body Part

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

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.

6.2.23 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.

This class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 6-21.

Table 6-21 Attributes Specific to USA Nationally Defined Body Part

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

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.

6.2.24 Videotex Body Part

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

This class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 6-22.

Table 6-22 Attributes Specific to Videotex Body Part

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

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 6.3.9, “Videotex Syntax,” on page 6-27.

6.3 IM Syntax Definitions

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

6.3.1 Acknowledgment Mode

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

Table 6-23 Acknowledgement Mode Values

automatic	The originator authorizes RNs in blanket fashion
manual	The originator authorizes the RN individually

6.3.2 Discard Reason

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

Table 6-24 Discard Reason Values

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.

6.3.3 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:

Table 6-25 IA5 Repertoire Values

IA5	The full IA5 character set (which is nearly identical to ASCII).
ITA2	The ITA2 (i.e., Telex) character set.

6.3.4 Importance

An instance of enumeration syntax Importance identifies the importance that an IPM's Authorizing 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.

6.3.5 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:

Table 6-26 ISO 6937 Repertoire Values

graphic	The full graphic repertoire.
part1-and-part2	
teletex	The Teletex subrepertoire.

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

6.3.6 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:

Table 6-27 Non-receipt Reason Values

ipm-auto-forwarded	The IPM is automatically forwarded.
ipm-discarded	The IPM is discarded.

6.3.7 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:

Table 6-28 Notification Reason Values

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.

6.3.8 Sensitivity

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

Table 6-29 Sensitivity Values

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.

6.3.9 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:

Table 6-30 Videotex Syntax Values

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.

EDI Messaging Package 7

7.1 EDIM Class Hierarchy

This section depicts the hierarchical organization of the EDI classes. Subclassification is indicated by indentation. The names of abstract classes are in *italics*. For example, **Negative Notification** is an immediate subclass of *EDI Notification*, an abstract class. The service is not required to support the `OM-Encode` function for any classes in the EDI Package. The `OM-Create` function must be supported for all concrete classes.

Object (defined in the *Solstice XOM Programming Reference Manual*)

- Authorization Information
- *Content* (defined in Chapter 5, "Message Handling Packages")
  - — *EDI Notification*
    - — Forwarded Notification
    - — Negative Notification
    - — Positive Notification
  - — EDIM
- Cross Referencing Information

- Delivery Envelope (defined in Chapter 5, “Message Handling Packages”)
- EDI Application Security Elements
- EDI Extension
- EDI Heading
- EDI Notification Requests
- EDIM Externally Defined Body Part
- EDIM Identifier
- EDIN Receiver
- Externally Defined Body Part (defined in Chapter 6, “Interpersonal Messaging Packages”)
- Integrity Check Basis
- Interchange Sender-recipient
- OR Name (defined in Chapter 5, “Message Handling Packages”)
- *Primary Body Part*
  - EDI Body Part
  - EDIM Body Part
- *Reason Code*
  - *FN Reason Code*
    - FN PDAU Reason Code
    - FN UAMS Reason Code
    - FN User Reason Code
  - *NN Reason Code*
    - NN PDAU Reason Code
    - NN UAMS Reason Code
    - NN User Reason Code
- Recipient Reference
- Recipients
- Security Elements
- Service String Advice
- Syntax Identifier

7.2 EDIM Class Definitions

This section defines the classes of the EDI Package. The object identifier associated with this package is

{joint-iso-ccitt mhs-motis(6) group(6) white(1) api(2) edi(7) edi-pkg(1)}

(with the encoding **\\x56\\x06\\x01\\x02\\x07\\x01**).

This object identifier is represented by the constant `EDI_PACKAGE` *{*`EDI_PACKAGE`*}*. Each OM class is described in a separate section, which describes the OM attributes specific to that OM class. The OM attributes that may be found in an instance of an OM class are those OM attributes specific to that OM class and those inherited from each of its superclasses. The OM attributes specific to an OM class are defined in a table. The table gives the name of each OM attribute, the syntax of each of its values, any restrictions upon the length of each value, any restrictions upon the number of values, and the value, if any, that the `OM-Create` function supplies.

The attribute descriptions are derived directly from the EDI Messaging System standards. It is not the intent of this specification to add any additional constraining semantics to the information elements described. In cases of ambiguity or conflict in the semantics of attributes, the EDI messaging standards are to be taken as definitive. This document provides a mapping of the structure of information objects of the standards into a form suitable for use with APIs and in particular with the Message Access, Message Transfer and Object Management Interfaces.

7.2.1 Authorization Information

An instance of class **Authorization Information** indicates who authorized the interchange. This class is semantically identical to the “Authorization information” in the ANSIX12 Interchange.

An instance of this class has the attributes of its superclass (*Object*) and additionally the attributes listed below.

Table 7-1 Attributes Specific to Authorization Information

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
Authorization Information	String(Teletex)	1-10	1	-
Authorization Information Qualifier	String(Teletex)	1-2	0-1	-

Further details on the meaning of this class and its attributes can be found in the ANSIX12 standard.

7.2.2 Content

The **Content** class is defined in the Message Handling (MH) Package in Section 5.2.8, “Content,” on page 5-12. The EDI Package closure contains this class and its closure.

An instance of class **Content** is the information that a message is intended to convey to its recipients. The purpose of this abstract class is to provide a common superclass for content type-dependent classes defined by different messaging applications. The value of the OM attribute, Content, of the Message class (in the MH package) may be an instance of any concrete subclass of the Content class. Examples of such subclasses include *General Content* (defined in the MH Package), *Interpersonal Message* and *Interpersonal Notification* (defined in the IM Package), *EDIM* and *EDI Notification* (defined in this chapter), and message content type-dependent classes that may be defined in other specifications. The value of the Content Type attribute of the Message class indicates the syntax and semantics of the Content attribute. Similarly, the Content Type attribute of the Probe class (also part of the MH Package) indicates the syntax and semantics of the message content to which the probe refers. Section 5.2.8, “Content,” on page 5-12 defines values of the Content Type attribute for use with Interpersonal Messaging. The content type associated with EDI is represented by the constant `EDI_CONTENT_TYPE` *{*`EDI_CONTENT_TYPE`*}*. This content type is used with both EDI Notifications and EDIMs.

7.2.3 Cross Referencing Information

An instance of class **Cross Referencing Information** contains a reference to a body part. The referenced body part may be in the same EDIM containing the Cross Referencing Information instance, or within other EDIMs.

An instance of this class has the attributes of its superclass (*Object*) and additionally the attributes listed below.

Table 7-2 Attributes Specific to Cross Referencing Information

OM Attribute	**Value Syntax**	**Value Length**	**Value Number**	**Value Initially**
Application Cross Reference	String(Octet)	-	1	-
Message Reference	Object(EDIM Identifier)	-	0-1	-
Body Part Reference	Integer	-	1	-

Application Cross Reference
Application-specific information about the referenced body part.

Message Reference
Indicates the EDIM containing the referenced body part. If this attribute is absent, the message referred to is the EDIM containing the Cross Referencing Information instance.

Body Part Reference
Indicates the referenced body part. See the **EDIM Externally Defined Body Part** class.

7.2.4 Delivery Envelope

The **Delivery Envelope** class is defined in the Message Handling Package in Section 5.2.15, “Delivery Envelope,” on page 5-17. The EDI Package closure contains this class and its closure.

7.2.5 EDI Application Security Elements

An instance of class **EDI Application Security Elements** identifies a set of security elements. This class is used to allow an EDI application to exchange security elements having an end-to-end significance. An instance of this class has the attributes of its superclass *(Object*) and additionally the attributes listed below.

Table 7-3 Attributes Specific to EDI Application Security Elements

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
EDI Application Security Element	String(Bit)	0-8191	0-1	-
EDI Encrypted Primary Body Part	Boolean	-	0-1	-
EDI Application Security Extensions	Object(EDI Extension)	-	0 or more	-

EDI Application Security Element
Indicates the Application Security Elements referred to by this instance.

EDI Encrypted Primary Bodypart
Indicates whether the associated primary bodypart is encrypted.

EDI Application Security Extensions
Contains extensions to the EDI Application Security Elements class.

7.2.6 EDI Body Part

An instance of class **EDI Body Part** carries a single EDI Interchange. The reference definition of an EDI Interchange used here is that of EDIFACT (ISO 9735) (see Appendix C, "Referenced Documents"). An instance of this class has the attributes of its superclass (*Object*) and additionally the attributes listed below.

Table 7-4 Attributes Specific to EDI Body Part

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
Body	String(Octet)	-	1	-

Body
Contains an octet string encoding the EDI Interchange.

7.2.7 EDI Extension

An instance of class **EDI Extension** provides a mechanism for inclusion of future extensions to the EDI messaging standards (see *EDI Messaging Systems*, in Appendix C, "Referenced Documents") in information classes defined in this document.

An instance of this class has the attributes of its superclass *(Object*) and additionally the attributes listed below.

Table 7-5 Attributes Specific to EDI Extension

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
Criticality	Boolean	-	1	False
Extension Type	String(Object Identifier)	-	1	-
Extension Value	any	-	0-1	-

Criticality
Indicates whether the extension is critical for acceptance of Responsibility. An extension marked as non-critical (criticality set to FALSE) may be ignored or discarded, while an extension marked as critical must be known and performed for acceptance of Responsibility of an EDIM.

Extension Type
Indicates the type of the extension.

Extension Value
Indicates the value of the extension. The value must be of the type indicated by the value of the Extension Type attribute.

7.2.8 EDI Heading

An instance of class **EDI Heading** is a set of characteristics describing an EDIM.

An instance of this class has the attributes of its superclass *(Object*) and additionally the attributes listed below.

Table 7-6 Attributes Specific to EDI Heading

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
This EDIM	Object(EDIM Identifier)	-	1	-
Originator	Object(OR Name)	-	0-1	-
Recipients	Object(Recipients)	-	0 or more	-
EDIN Receiver	Object(EDIN Receiver)	-	0-1	-
Responsibility Forwarded	Boolean	-	1	False
EDI Body Part Type	String(Object Identifier)	-	1	edifact-ISO646
Incomplete Copy	Boolean	-	1	False
Expiry Time	String(UTC Time)	-	0-1	-
Related EDI Messages	Object(EDIM Identifier)	-	0 or more	-
Related non-EDI Messages	Object(External)	-	0 or more	-
Obsoleted EDIMs	Object(EDIM Identifier)	-	0 or more	-
EDI Application Security Elements	Object(EDI Application Security Elements)	-	0-1	-
Cross Referencing Information	Object(Cross Referencing Information)	-	0 or more	-
EDI Message Type	String(Teletex)	1-6	0 or more	-
Service String Advice	Object(Service String Advice)	-	0-1	-
Syntax Identifier	Object(Syntax Identifier)	-	0-1	-
Interchange Sender	Object(Interchange Sender-Recipient)	-	0-1	-
Date And Time of Preparation	String(UTC Time)	-	0-1	-
Application Reference	String(Teletex)	1-14	0-1	-
Heading Extensions	Object(EDI Extension)	-	0 or more	-

This EDIM
Identifies the EDIM.

Originator

The OR Name that identifies the EDIM's originator. If the Originator attribute is not present in the EDIM Heading on reception, then the Originating-name of the delivery envelope shall be used to determine the originator of the EDIM (see *CCITT X.411 | ISO/IEC 10021-4*, listed in Appendix C, "Referenced Documents). The Originating-name is contained in the Originator Name attribute of an instance of the class **Delivery Envelope** defined as part of the MH Package in Section 5.2.15, "Delivery Envelope," on page 5-17.

Recipients

Identifies the user(s) and distribution lists (DL) that are the preferred recipient(s) of the EDIM. The values of this attribute are instances of the Recipient class, one instance for each recipient. If the Recipients attribute is not present in the EDIM Heading on reception, then the This-recipient-name of the delivery envelope shall be used to determine the recipient of the EDIM (see *CCITT X.411 | ISO/IEC 10021-4*). The This-recipient-name is contained in the Intended Recipient Name attribute of an instance of the class **Delivery Envelope** defined as part of the MH Package in Section 5.2.15, "Delivery Envelope," on page 5-17. A message can be redirected or forwarded so the preferred recipients indicated in this attribute are not necessarily the final recipients of the EDIM.

EDIN Receiver

Identifies the recipient to whom EDINs are to be sent. The value of this attribute is supplied by the originator of the EDIM when the Recipient of a requested notification is different from the Originator of the message. It consists of a sequence of Recipient Name, **EDIM Identifier** and First Recipient (see the definition of the **EDIN Receiver** class). This attribute shall not be present if **EDI Notification Requests** are not made. (**EDI Notification Requests** are made within the value of the Recipients attribute.)

This attribute shall be present in a forwarded message when the forwarding EDI user agent (EDI-UA) or EDI message store (EDI-MS) forwards Responsibility. This attribute may be present when the forwarding EDI-UA accepts Responsibility. Rules related to the construction of this attribute are given in the EDI messaging standards (see Appendix C, "Referenced Documents").

Responsibility Forwarded

Indicates whether Responsibility was forwarded. If this attribute has the value TRUE it indicates to a receiving UA that Responsibility was forwarded. If this attribute has the value FALSE (or is absent) it indicates to a receiving UA, that the security elements of the inner envelope have been checked. Subject to the security policy in force, the security elements may have been checked when the message was forwarded. However, when Responsibility is accepted, the security elements shall be checked.

EDI Body Part Type

Indicates the EDI standard and EDI character sets used in the **Primary Body Part**. It is represented by a single object identifier. The following standard values have object identifiers defined in the EDI messaging standards (see Appendix C, "Referenced Documents"). The first column of the table lists the symbolic values. The second column specifies, in ASN.1, the object identifiers denoted by the values.

Table 7-7 Selected Values of the EDI Body Part Type Attribute

Value	Object Identifier (ASN.1)
edifact-ISO646	{joint-iso-ccitt mhs-motis(6) edims(7) 11 0}
edifact-T61	{joint-iso-ccitt mhs-motis(6) edims(7) 11 1}
edifact-octet	{joint-iso-ccitt mhs-motis(6) edims(7) 11 2}
ansiX12-ISO646	{joint-iso-ccitt mhs-motis(6) edims(7) 11 3}
ansiX12-T61	{joint-iso-ccitt mhs-motis(6) edims(7) 11 4}
ansiX12-octet	{joint-iso-ccitt mhs-motis(6) edims(7) 11 5}
ansiX12-ebcdic	{joint-iso-ccitt mhs-motis(6) edims(7) 11 6}
untdi-ISO646	{joint-iso-ccitt mhs-motis(6) edims(7) 11 7}
untdi-T61	{joint-iso-ccitt mhs-motis(6) edims(7) 11 8}
untdi-octet	{joint-iso-ccitt mhs-motis(6) edims(7) 11 9}
private-octet	{joint-iso-ccitt mhs-motis(6) edims(7) 11 10}
undefined-octet	{joint-iso-ccitt mhs-motis(6) edims(7) 11 11}

The default value of this attribute is edifact-ISO646.

Both the **EDI Body Part**, and those values within an instance of an EDI Heading that are Octet Strings and are derived from the EDI Interchange, are encoded in the character set referred to by this attribute.

The value of the **EDI Body Part** Type attribute shall be used with the *EITs* (Encoded Information Types) class defined in the MH Package. This enables a UA to signal to the MTS the EDI standard to which the EDIM's **Primary Body Part** complies. The MTS shall make use of this information, if the recipient UA has registered delivery restrictions on Encoded Information Types, to decide if it can deliver the EDIM.

The term Encoded Information Type is defined in CCITT X.402 | ISO / IEC 10021-2. See also CCITT X.411 | ISO / IEC 10021-4 (see Appendix C, "Referenced Documents".

Incomplete Copy
The Incomplete Copy attribute indicates that the forwarded EDIM is an incomplete copy of an EDIM. This attribute shall have the value TRUE if body parts are removed when an EDIM is forwarded. The default value is FALSE.

The term *Forwarded EDIM* is defined in the EDI messaging standards (see Appendix C, "Referenced Documents").

Expiry Time
The date and time when the originator considers this EDIM to be no longer valid.

Related EDI Messages
Identifies EDIMs that the originator of this EDIM considers related to it. If the related message identifies messages from other services, the user component of the message identifier (EDIM Identifier) must be present. Message identifier values of the referenced message of other service types than EDIMG are carried in this attribute.

Related Non-EDI Messages
Identifies messages (other than EDIMs) that the originator of this EDIM considers related to it. A related message may be an EDIM or any other type of message (for example, an IPM).

Obsoleted EDIMs
Identifies one or more EDIMs that the present EDIM makes obsolete. Each obsolete EDIM is identified by an EDIM Identifier.

EDI Application Security Elements
Allows an EDI application to exchange security elements having an end-to-end significance.

Cross Referencing Information
Allows an EDI application to reference individual body parts within the same EDIM and within other EDIMs. It contains a set of cross reference data.

EDI Message Type
Indicates the EDI Message type(s) present in the EDI Interchange. The values of this attribute are distinct strings identifying the message types defined in the EDI standards. The values for this attribute shall be:

- EDIFACT—Message Type from the UNH segment
- ANSIX12—Transaction Set ID from the ST segment
- UNTDI—Message Type from the MHD segment

Service String Advice
Indicates the Service String Advice of the EDI Interchange. This is semantically identical to the "UNA, Service string advice" of the EDIFACT Interchange. The meaning of this attribute is described fully in the EDIFACT standard (see Appendix C, "Referenced Documents").

Syntax Identifier
Indicates the syntax used in this EDI Interchange. This is semantically identical to the "Syntax identifier" of the EDIFACT UNB segment (see Appendix C, "Referenced Documents").

Interchange Sender
Indicates the sender of the EDI Interchange. This is semantically identical to the "Interchange sender" of the EDIFACT UNB segment (see Appendix C, "Referenced Documents").

Date And Time of Preparation
Indicates the date and time of preparation of the EDIM. This is in UTC Time and is derived from the "Date and time of preparation" of the EDIFACT UNB segment (see Appendix C, "Referenced Documents").

Application Reference
Provides a general reference to an application or function. This is semantically identical to the "Application reference" segment of the EDIFACT UNB segment (see Appendix C, "Referenced Documents").

Heading Extensions
Contains extensions to the EDI Heading class.

7.2.9 EDI Notification

An instance of class **EDI Notification** is an information object conveyed between users of EDI Messaging. The information objects that users exchange in EDI Messaging are of two kinds: EDI Messages (EDIM) and EDI Notifications (EDIN). The *EDI Notification* class is an abstract class for which three concrete subclasses are defined in this specification: **Forwarded Notification (FN)**, **Negative Notification (NN)**, and **Positive Notification (PN)**.

The recipient of the EDIN is the Originator of the subject EDIM, or, if present, the OR Name indicated in the EDIN Receiver attribute of the EDIM. There shall be at most one recipient specified for an EDIN. There shall be at most one PN, NN or FN originated for each subject EDIM by each recipient of whom notifications are requested (except that an NN may be originated by the same UA subsequent to an FN, in accordance with the EDI messaging standards). One FN is originated, if, and only if, it is requested by each recipient that forwards an EDIM. In accordance with the provisions of the EDI Messaging standards (see Appendix C, "Referenced Documents"), the originator of the subject EDIM shall receive at most one PN or NN for each recipient of whom notifications were requested, regardless of how many times the EDIM is forwarded. The originator may, however, receive multiple FNs.

This class is an abstract class which has the attributes of its superclass (*Object*) and additionally the attributes listed below.

Table 7-8 Attributes Specific to EDI Notification

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
Subject EDIM	Object(EDIM Identifier)	-	1	-
EDIN Originator	Object(OR Name)	-	1	-
First Recipient	Object(OR Name)	-	0-1	-
Notification Time	String(UTC Time)	-	1	-
Security Elements	Object(Security Elements)	-	0-1	-

Table 7-8 Attributes Specific to EDI Notification

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
EDIN Initiator	Enum(EDIN Initiator)	-	1	-
Notification Extensions	Object(EDI Extension)	-	0 or more	-
Supplementary Information	String(Teletex)	1-256	0-1	-

Subject EDIM
Identifies the EDIM to which this EDIN refers. If Responsibility has been forwarded, this identifier is passed in the EDIN Receiver attribute of the Heading of the EDIM, otherwise it is passed in the This EDIM attribute.

EDIN Originator
Contains the OR Name of the UA constructing the notification. The class OR Name is defined in the MH Package.

First Recipient
Contains the OR Name of the first recipient in a forwarding chain. The class OR Name is defined in the MH Package. This attribute, together with other attributes, is used by the recipient of the notification to correlate the notification and the original message. If the originator of the EDIN is not the recipient specified by the originator of the subject EDIM, then the First Recipient attribute shall be present in the EDIN.

Notification Time
Contains the date and time at which this notification was generated.

Security Elements
Indicates the security services requested on this notification.

EDIN Initiator
Indicates who initiated the notification. Each value is one of:

- **internal-ua**—The UA generated the EDIN either for local reasons or because the generation had been delegated to it by the user.
- **internal-ms**—The MS generated the EDIN either for local reasons or because the generation had been delegated to it by the user
- **external-ua**—The generation of the EDIN was requested by the user.

Origination of a **Positive Notification** implies that Responsibility has been accepted, regardless of the value of this attribute. The value of this attribute shall be consistent with the choice (UA, user, PDAU) of the Reason Code attribute for **Negative Notifications** and **Forwarded Notifications**.

Notification Extensions
Contains extensions to the EDI Notification class.

Supplementary Information
The Supplementary Information attribute may be used to return further information to the EDIN recipient to clarify the notification.

7.2.10 EDI Notification Requests

An instance of class **EDI Notification Requests** describes the type of notifications and the security required for the notifications which are requested of a preferred recipient.

An instance of this class has the attributes of its superclass *(Object*) and additionally the attributes listed below.

Table 7-9 Attributes Specific to EDI Notification Requests

OM Attribute	Value Syntax[1]	Value Length	Value Number	Value Initially
Positive Notification	Boolean	-	1	False
Negative Notification	Boolean	-	1	False
Forwarded Notification	Boolean	-	1	False
Proof of Notification	Boolean	-	1	False
Non-repudiation of Notification	Boolean	-	1	False
Proof of Reception	Boolean	-	1	False
Non-repudiation of Reception	Boolean	-	1	False

[1] The Bit String syntax of the EDI Messaging standards (see Appendix C, "Referenced Documents) has been mapped into Booleans in the interface class definition.

Each of the attributes in this class corresponds to a request that may be made of the preferred recipient. All attributes default to False, indicating that the request is not made.

Positive Notification
Requests a Positive Notification (PN), that reports its originator's acceptance of Responsibility of an EDIM.

Negative Notification
Requests a Negative Notification (NN), that reports its originator's refusal to accept Responsibility of an EDIM.

Forwarded Notification
Requests a Forwarded Notification (FN), that reports that Responsibility of an EDIM has been forwarded together with the EDIM.

Proof of Notification
Indicates whether content-integrity-check shall be requested when submitting the EDIN to the MTS. (See *CCITT Recommendation X.411 | ISO/IEC 10021-4*.)

Non-repudiation of Notification
Indicates whether content-integrity-check shall be requested with a non-repudiable certificate when submitting the EDIN to the MTS. (See *CCITT Recommendation X.411 | ISO/IEC 10021-4*.)

Proof of Reception
Indicates whether, when submitting the EDIN to the MTS, content-integrity-check (possibly in the message token), or the message-origin-authentication-check (depending on the security policy in force) shall be requested. A notification shall contain the security elements and shall be signed on submission to the MTS, using content-integrity-check (possibly in the message token) or message-origin-authentication-check (depending on the security policy in force) as defined in *CCITT X.411 | ISO/IEC 10021-4*.

Non-repudiation of Reception
Indicates whether when submitting the EDIN to the MTS, a non-repudiable content-integrity-check (possibly in the message token) or a message-origin-authentication-check (depending on the security policy in force) shall be requested.

Note – There are no security services available in this implementation of the MTS.

7.2.11 EDIM

An instance of class **EDIM** is an information object conveyed between users in EDI Messaging. The information objects that users exchange in EDI messaging are of two kinds: EDI Messages (EDIM) and EDI Notifications (EDIN).

An instance of this class has the attributes of its superclass *(Object*) and additionally the attributes listed below.

Table 7-10 Attributes Specific to EDIM

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
Heading	Object(EDI Heading)	-	1	-
Primary Body Part	Object(Primary Body Part)	-	1	-
Additional Body Parts	Object(EDIM Externally Defined Body Part)	-	0 or more	-

Heading
A set of heading attributes, each an information item that gives a characteristic of the EDI Message.

Primary Body Part
Contains an EDI information object. Examples of types of EDI information objects are EDI Interchanges defined by EDIFACT, UNTDI and ANSIX12 (see Appendix C, "Referenced Documents").

Additional Body Parts
Additional body parts related to the **Primary Body Part** but of different types. Examples of related body parts include textual information, voice annotation or graphics to be used in conjunction with the interchange.

7.2.12 EDIM Body Part

An instance of class **EDIM Body Part** contains an EDIM, and optionally, its delivery envelope. This body part is used for forwarding EDIMs. When an EDIM is forwarded, its structure shall comply with the rules given in the EDI messaging standards (see Appendix C, "Referenced Documents").

An instance of this class has the attributes of its superclass *(Object*) and additionally the attributes listed below.

Table 7-11 Attributes Specific to EDIM Body Part

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
Delivery Envelope	Object(Delivery Envelope)	-	0-1	-
EDI Supplementary Info.	String(Teletex)	1-256	0-1	-
Heading	Object(EDI Heading)	-	1	-
Primary Body Part	Object(Primary Body Part)	-	0-1	-
Additional Body Parts	Object(EDIM Ext. Defined Body Part)	-	0 or more	-

Delivery Envelope
The delivery envelope of the EDIM. The Delivery Envelope class is part of the Message Handling Package. A value of the Delivery Envelope attribute must be an instance of the Delivery Envelope class as specified in the MH Package with the exception that the MTS Identifier attribute shall not be present and the Delivery Time attribute is optional (that is, it may not be present).

The delivery envelope shall be present if security services are invoked.

EDI Supplementary Information
Additional parameters specific to EDI.

Heading
Heading of the EDIM.

Primary Body Part
The primary body part of the EDIM. This attribute may be absent if the primary body part has been removed from the EDIM.

Additional Body Parts
Additional body parts of the EDIM. This attribute may contain complete body parts or body parts from which the data portion has been removed. The latter, incomplete values, are used as place holders for removed body parts. They may consist of only the Body Part Reference, or a modified Externally Defined Body Part. In the latter case the object identifier and Body Part Reference of the removed body part are preserved; from the External Parameters attribute (if present) and External Data attribute of the

removed body part, only the object identifier and the identifier octets of the appropriate Encoding attribute of the External object instance are preserved. That is, in the instance of the External object, the Encoding attribute which is present shall be of zero length and hence, have no content. (Refer to the definition of the External class in the *Object Management Package*.)

7.2.13 EDIM Externally Defined Body Part

An instance of class **EDIM Externally Defined Body Part** contains body parts that relate to the **Primary Body Part** and may be carried together with an EDI Interchange. These body parts are not EDI Interchanges and do not include EDI Interchanges. Additional body parts are externally defined and represent information objects whose semantics and abstract syntax are denoted by an object identifier which the body part carries. They have Parameters and Data components and optionally a Body Part Reference that may be used for cross referencing to a body part.

An instance of this class has the attributes of its superclass *(Object*) and additionally the attributes listed below.

Table 7-12 Attributes Specific to EDIM Externally Defined Body Part

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
Body Part Reference	Integer	-	0-1	-
External Body Part	Object(Externally Defined Body Part)	-	0-1	-

Body Part Reference
A reference to this body part. The reference may be used for cross referencing body parts. The references assigned to body parts shall be unique within an EDIM. They are assigned when a body part is created, and are not modified subsequently. A reference shall be present if the originator wishes to cross reference the body part at creation or in the future.

External Body Part
The content of the body part. The value of this attribute is an instance of the class Externally Defined Body Part which is part of the Interpersonal Messaging Package defined in Chapter 6, “Interpersonal Messaging Packages”. Some Externally Defined body part types are defined in *CCITT X.420 | ISO/IEC 10021-7* (see Appendix C, “Referenced Documents”).

7.2.14 EDIM Identifier

An instance of class **EDIM Identifier** identifies an EDIM. It contains an OR Name and a string which may contain a time, sequence number, or other information sufficient to uniquely identify an EDIM.

An instance of this class has the attributes of its superclass (*Object*) and additionally the attributes listed below.

Table 7-13 Attributes Specific to EDIM Identifier

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
User	Object(OR Name)	-	1	-
User-relative-identifier	String(Printable)	0-64[1]	1	-

[1] A length of zero is discouraged.

User
Identifies the user who originates the EDIM. One of the user's OR Names. The class OR Name is defined in the MH Package.

User-relative-identifier
Unambiguously identifies the EDIM, distinguishing it from all other EDIMs that the user originates.

7.2.15 EDIN Receiver

An instance of class **EDIN Receiver** identifies the recipient to whom EDINs are to be sent. An instance is created by the originator of the EDIM when the Recipient of a requested notification is different from the Originator of the message.

An instance of this class has the attributes of its superclass (*Object*) and additionally the attributes listed below.

Table 7-14 Attributes Specific to EDIN Receiver

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
EDIN Receiver Name	Object(OR Name)	-	1	-
Original EDIM Identifier	Object(EDIM Identifier)	-	0-1	-
First Recipient	Object(OR Name)	-	0-1	-

EDIN Receiver Name
Identifies the recipient of a requested notification. The class OR Name is defined in the MH Package.

Original EDIM Identifier
Identifies the original EDIM for which the notification is being sent.

First Recipient
Identifies the first recipient in a forwarding chain. This attribute (together with information from instances of other classes), is used by the recipient of the notification to correlate the notification and the original message. The First Recipient attribute shall not be present if the corresponding EDIM contains more than one recipient with associated **EDI Notification Requests**. The class OR Name is defined in the MH Package.

The Original EDIM Identifier and the First Recipient attributes shall not be present when the Primary Body Part is an EDI Body Part (that is, when the original originator first creates the EDIM). The Original EDIM Identifier and First Recipient attributes are included in order to allow the recipient to construct the EDIN for a forwarded EDIM.

7.2.16 Externally Defined Body Part

The **Externally Defined Body Part** class is defined in the Interpersonal Messaging Package in Section 6.2.4, “Externally Defined Body Part,” on page 6-4. The EDI Package closure contains this class and its closure.

7.2.17 FN PDAU Reason Code

An instance of class **FN PDAU Reason Code** indicates why the Responsibility of the subject EDIM was forwarded. An instance of this class contains reasons indicated by a PDAU for the issuing of an FN.

An instance of this class has the attributes of its superclasses *(Reason Code* and *Object*) and no additional attributes. A set of standard values for the Reason Code attributes are defined below. These standard values are specific to this subclass of Reason Code.

Basic Code
The following standard values are defined:

- **unspecified**—The reason has not been specified by the PDAU.
- **forwarded-for-physical-rendition-and-delivery**—The PDAU has determined that it can render the EDIM for physical delivery.

7.2.18 FN Reason Code

An instance of class **FN Reason Code** indicates why the Responsibility of the subject EDIM was forwarded. This class is an abstract class which has the attributes of its superclasses (*Reason Code* and *Object*) and no other attributes. This specification defines three subclasses of **FN Reason Code**, for reason codes issued from an EDI-UA or EDI-MS (**FN UAMS Reason Code**), from a user (**FN User Reason Code**), or from a physical delivery access unit, PDAU (**FN PDAU Reason Code**).

7.2.19 FN UAMS Reason Code

An instance of class **FN UAMS Reason Code** indicates why the Responsibility of the subject EDIM was forwarded. An instance of this class contains reasons indicated by a UA or MS for the issuing of a forwarded notification.

An instance of this class has the attributes of its superclasses (*Reason Code* and *Object*) and additionally the attributes listed below.

Table 7-15 Attributes Specific to FN UAMS Reason Code

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
FN Security Check	Boolean	-	1	False

Basic Code
This attribute is inherited from the superclass Reason Code. It has the following standard values:

- unspecified (used when the reason has not been specified by the UA or MS)
- onward-routing (used when the UA decides to re-route the subject EDIM for local reasons)
- recipient-unknown
- originator-unknown
- forwarded-by-edi-ms

Diagnostic
This attribute is inherited from the superclass Reason Code. It has the following standard values:

- recipient-name-changed
- recipient-name-deleted

FN Security Check
Indicates whether all security features present have been validated.

7.2.20 FN User Reason Code

An instance of class **FN User Reason Code** indicates why the Responsibility of the subject EDIM was forwarded. An instance of this class contains reasons indicated by a User for the issuing of an FN.

An instance of this class has the attributes of its superclasses (*Reason Code* and *Object*) and no additional attributes. A set of standard values for the Reason Code attributes are defined below. These standard values are specific to this subclass of Reason Code.

Basic Code

The following standard values are defined:

- unspecified (used when the reason has not been specified by the user)
- forwarded-for-archiving
- forwarded-for-information
- forwarded-for-additional-action
- subscription-changed
- heading-field-not-supported
- bodypart-type-not-supported
- message-type-not-supported
- syntax-identifier-not-supported
- interchange-sender-unknown
- user-defined-reason

7.2.21 Forwarded Notification

An instance of class **Forwarded Notification** reports that Responsibility of an EDIM has been forwarded together with the subject EDIM. A **Forwarded Notification** (FN) is sent by a UA, if, and only if, the originator has requested forwarding notification. A **Forwarded Notification** is sent when the UA determines that it cannot accept Responsibility, and decides to forward the **EDIM**, and the **EDI Notification Requests** contained in the **EDIM**, to another UA.

An instance of this class has the attributes of its superclasses *(EDI Notification*, and *Object*) and additionally the attributes listed below.

Table 7-16 Attributes Specific to Forwarded Notification

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
Forwarded To	Object(OR Name)	-	1	-
FN Reason Code	Object(FN Reason Code)	-	1	-

Forwarded To

Indicates the new recipient of the (forwarded) subject EDIM.

FN Reason Code

The Forwarded Notification Reason indicates why the Responsibility of the subject EDIM was forwarded. Additional information may be carried in any combination of the Reason Code attribute or the Supplementary Info attribute.

7.2.22 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.

An instance of this class has the attributes of its superclass *(Object*) and additionally the attributes listed below.

Table 7-17 Attributes Specific to Integrity Check Basis

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
Algorithm	Object(Algorithm)	-	1	-
Content	String(Octet)	-	1	-

Further details on the meaning of this class and its attributes can be found in the EDIFACT standard (see Appendix C, "Referenced Documents").

Algorithm

Identifies the algorithm used to compute the integrity check.

Content

The unencrypted content whose integrity is being checked.

7.2.23 Interchange Sender-recipient

An instance of class **Interchange Sender-Recipient** indicates the sender or recipient of an EDI Interchange and is semantically identical to the "Interchange sender / Interchange recipient" of the EDIFACT UNB segment.

An instance of this class has the attributes of its superclass *(Object*) and additionally the attributes listed below.

Table 7-18 Attributes Specific to Interchange Sender-Recipient

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
Identification Code	String(Teletex)	1-35	1	-
Identification Code Qualifier	String(Teletex)	1-4	0-1	-
Routing Address	String(Teletex)	1-14	0-1	-

Further details on the meaning of this class and its attributes can be found in the EDIFACT standard (see Appendix C, "Referenced Documents").

Identification Code
Identifies the sender or recipient of an interchange and is semantically identical to the "Sender identification / Recipient identification" component of the Interchange sender / recipient of the EDIFACT UNB segment.

Identification Code Qualifier
The Identification Code Qualifier, if present, is a qualifier of the Identification Code of a sender or recipient. The qualifier is semantically identical to the "Identification Code Qualifier" component of the Interchange sender / recipient of the EDIFACT UNB segment.

Routing Address
The Routing Address, if present, is an address for routing to the sender or recipient specified in the Identification Code. This is semantically identical to the "Address for reverse routing / Routing address" component of the Interchange sender / recipient of the EDIFACT UNB segment.

7.2.24 Negative Notification

An instance of class **Negative Notification** reports its originator's refusal to accept Responsibility of an EDIM. A **Negative Notification (NN)** is sent by a UA, if, and only if, the originator has requested negative notification. An NN is sent when the UA determines that it can neither accept Responsibility, nor forward the **EDIM** and the **EDI Notification Requests** contained in the EDIM to another UA.

An instance of this class has the attributes of its superclasses *(EDI Notification*, and *Object*) and additionally the attributes listed below.

Table 7-19 Attributes Specific to Negative Notification

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
NN Reason Code	Object(NN Reason Code)	-	1	-

NN Reason Code
The Negative Notification Reason indicates why the subject EDIM could not be passed to the user by the UA originating the EDIN. Additional information may be carried in any combination of the Reason Code attribute or the Supplementary Info attribute.

7.2.25 NN PDAU Reason Code

An instance of class **NN PDAU Reason Code** indicates why the subject EDIM could not be passed to the user by the UA originating the EDIN. An instance of this class contains reasons indicated by a PDAU for the issuing of an NN.

An instance of this class has the attributes of its superclasses *(Reason Code* and *Object*) and no additional attributes. A set of standard values for the Reason Code attributes are defined below. These standard values are specific to this subclass of Reason Code.

Basic Code

The following standard values are defined:

- unspecified (used when the reason has not been specified by the PDAU)
- undeliverable-mail (used when the PDAU has determined that it cannot perform physical delivery of the EDIM)
- physical-rendition-not-performed (used when the PDAU cannot perform the physical rendition of the EDIM)

Diagnostic

The following standard values are defined:

- physical-rendition-attributes-not-supported

Reasons why the mail was undeliverable by the PDAU include:

- physical-delivery-address-incorrect
- physical-delivery-office-incorrect-or-invalid
- physical-delivery-address-incomplete
- recipient-unknown
- recipient-deceased
- organization-expired
- recipient-refused-to-accept
- recipient-did-not-claim
- recipient-changed-address-permanently
- recipient-changed-address-temporarily
- recipient-changed-temporary-address
- new-address-unknown
- recipient-did-not-want-forwarding
- originator-prohibited-forwarding

7.2.26 NN Reason Code

An instance of class **NN Reason Code** indicates why the subject EDIM could not be passed to the user by the UA originating the EDIN. This class is an abstract class which has the attributes of its superclasses (*Reason Code* and *Object*) and no other attributes. This specification defines three subclasses of **NN Reason Code**, for reason codes issued from an EDI-UA or EDI-MS (**NN UAMS Reason Code**), from a user (**NN User Reason Code**), or from a physical delivery access unit, PDAU (**NN PDAU Reason Code**).

7.2.27 NN UAMS Reason Code

An instance of class **NN UAMS Reason Code** indicates why the subject EDIM could not be passed to the user by the UA originating the EDIN. An instance of this class contains reasons indicated by a UA or MS for the issuing of a negative notification.

An instance of this class has the attributes of its superclasses *(Reason Code* and *Object*) and no additional attributes. A set of standard values for the Reason Code attributes are defined below. These standard values are specific to this subclass of Reason Code.

Basic Code

The following standard values are defined:

- unspecified (used when the reason has not been specified by the UA or MS)
- cannot-deliver-to-user (used when the EDI Interchange cannot be passed on to the user (physical delivery errors are indicated by this code).)
- delivery-timeout (used when the EDI Interchange could not be passed on to the user within a specified time limit)
- message-discarded (used when the UA / MS discarded the message before handoff to user)
- subscription-terminated (used when the recipient's subscription terminated after delivery but before handoff to user)
- forwarding-error (used when EDI Forwarding was attempted, but failed)
- security-error (use when there was a security error)

Diagnostic

The following standard values are defined:

- protocol-violation (used if the UA detects a protocol error)

- edim-originator-unknown (used when the originator of the EDIM is unknown to the UA or MS)
- edim-recipient-unknown (used when the recipient of the EDIM is unknown to the UA or MS)
- edim-recipient-ambiguous (used when the EDIM recipients or originator are not valid)
- action-request-not-supported (used when the action requested by the recipient is not performed)
- edim-expired (used when the expiry date of the received EDIM occurred before the subject EDIM was successfully passed to the user or forwarded by the EDI-UA)
- edim-obsoleted (used when the EDIM Identifier of the received EDIM was contained in the Obsoleted EDIM attribute of a previously received EDIM)
- duplicate-edim (used when the same EDIM was received more than once from the same originator)
- unsupported-extension (used when the EDIM contains an extension which is not supported by the UA)
- incomplete-copy-rejected (used when the EDI-UA does not accept EDIMs with the Incomplete Copy Indication true)
- edim-too-large-for-application (used when the EDIM cannot be delivered to the user due to length constraints forwarding error diagnostic codes)
- forwarded-edim-not-delivered (used when a Non-Delivery Report was received for the forwarded EDIM)
- forwarded-edim-delivery-time-out (used when a Delivery Report is not received within a given period.
- forwarding-loop-detected (used when the UA received an EDIM which contained a previously forwarded EDIM)
- unable-to-accept-responsibility (used when the EDI-UA cannot accept or forward responsibility)

Interchange header diagnostic codes:

- interchange-sender-unknown (used when the UA does not recognize the interchange-sender of the EDI interchange)
- interchange-recipient-unknown (used when the UA cannot find a valid interchange recipient in the Recipient Specifier)
- invalid-heading-field (used when the interchange contains an invalid heading attribute)
- invalid-bodypart-type (used when the interchange contains an invalid body part type)
- invalid-message-type (used when the interchange contains an invalid message type)

- invalid-syntax-id (used when the interchange contains an invalid syntax identifier)

Depending on the security policy in force, the security error diagnostic codes may or may not be present.

The security error diagnostic codes are:

- message-integrity-failure
- forwarded-message-integrity-failure
- unsupported-algorithm
- decryption-failed
- token-error
- unable-to-sign-notification
- unable-to-sign-message-receipt
- authentication-failure
- security-context-failure
- message-sequence-failure
- message-security-labelling-failure
- repudiation-failure
- proof-of-failure

7.2.28 NN User Reason Code

An instance of class **NN User Reason Code** indicates why the subject EDIM could not be passed to the user by the UA originating the EDIN. An instance of this class contains reasons indicated by the user for the issuing of an NN.

An instance of this class has the attributes of its superclasses *(Reason Code* and *Object*) and no additional attributes. A set of standard values for the Reason Code attributes are defined below. These standard values are specific to this subclass of Reason Code.

Basic Code

The following standard values are defined:

- unspecified (used when the reason has not been specified by the user)
- syntax-error (used when the user has discovered a syntax error within the EDI Interchange)
- interchange-sender-unknown
- interchange-recipient-unknown (used when the UA cannot find a valid interchange recipient in the Recipient Specifier attribute of the interchange)

- invalid-heading-field
- invalid-bodypart-type
- invalid-message-type
- functional-group-not-supported
- subscription-terminated (used when the recipient's subscription has been terminated, unknown to the EDIMS-User service)
- no-bilateral-agreement
- user-defined-reason

7.2.29 Object

The **Object** class is defined in the OM (Object Management) Package. The EDI Package closure contains this class. All classes in this specification are subclasses of Object.

7.2.30 OR Name

The **OR Name** class is defined in the Message Handling Package in Section 5.2.38, "OR Name," on page 5-58. The EDI Package closure contains this class and its closure.

7.2.31 Positive Notification

An instance of class **Positive Notification** is an EDIN that reports its originator's acceptance of Responsibility of an EDIM. A **Positive Notification** (PN) is sent by the recipient UA if, and only if, the originator of the EDIM has requested positive notification. The exact procedures which constitute acceptance of Responsibility are a local matter. For example, the UA may construct the PN as soon as it passes the message to the user. Alternatively, it may wait for an external stimulus from the user indicating that the message has been accepted and a PN can therefore be sent by the UA.

An instance of this class has the attributes of its superclasses *(EDI Notification*, and *Object*) and no additional attributes.

7.2.32 Primary Body Part

An instance of class **Primary Body Part** contains an EDI information object. An EDIM contains exactly one instance of a concrete subclass of the **Primary Body Part** class.

This class is an abstract class which has the attributes of its superclass (*Object*) and no additional attributes.

7.2.33 Reason Code

An instance of class **Reason Code** indicates why a notification was issued. This specification defines subclasses of **Reason Code**, each of which defines a set of standard values for the reason code attributes which are specific to different situations.

This class is an abstract class which has the attributes of its superclass (*Object*) and additionally the attributes listed below.

Table 7-20 Attributes Specific to Reason Code

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
Basic Code	Integer	-	1	-
Diagnostic	Integer	-	0-1	-

Basic Code
Indicates the basic reason a notification is sent. The value must be between 0 and 32767.

Diagnostic
This attribute is used to further specify the error signalled in the basic code. The value must be between 1 and 32767, if this attribute is present. Additional information may be indicated in the supplementary information attribute of the notification.

7.2.34 Recipient Reference

An instance of class **Recipient Reference** identifies a reference meaningful to the recipient's EDI application. This class is semantically identical to the "Recipient's Reference, Password" of the EDIFACT UNB segment.

An instance of this class has the attributes of its superclass *(Object*) and additionally the attributes listed below.

Table 7-21 Attributes Specific to Recipient Reference

OM Attribute	**Value Syntax**	**Value Length**	**Value Number**	**Value Initially**
Recipient Reference	String(Teletex)	1-14	1	-
Recipient Reference Qualifier	String(Teletex)	1-2	0-1	-

Further details on the meaning of this class and its attributes can be found in the EDIFACT standard (see Appendix C, "Referenced Documents").

7.2.35 Recipients

An instance of class **Recipients** identifies the user(s) and distribution lists (*DL*) that are the (preferred) recipient(s) of the EDIM. This class also contains attributes used to make certain requests of the preferred recipients including requests for notifications and acknowledgements.

An instance of this class has the attributes of its superclass (*Object*) and additionally the attributes listed below.

Table 7-22 Attributes Specific to Recipients

OM Attribute	**Value Syntax**	**Value Length**	**Value Number**	**Value Initially**
Recipient	Object(OR Name)	-	1	-
Action Request	String(Object Identifier)	-	1	for-action
EDI Notification Requests	Object(EDI Notification Requests)	-	0-1	-
Responsibility Passing Allowed	Boolean	-	1	False
Interchange Recipient	Object(Interchange Sender-Recipient)	-	0-1	-
Recipient Reference	Object(Recipient Reference)	-	0-1	-

Table 7-22 Attributes Specific to Recipients

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
Interchange Control Reference	String(Teletex)	1-14	0-1	-
Processing Priority Code	String(Teletex)	1	0-1	-
Acknowledgement Request	Boolean	-	1	False
Comms Agreement Id	String(Teletex)	1-35	0-1	-
Test Indicator	Boolean	-	1	False
Authorization Information	Object(Authorization Information)	-	0-1	-
Recipient Extensions	Object(EDI Extension)	-	0 or more	-

Recipient
Identifies a preferred recipient. The class OR Name is defined in the MH Package.

Action Request
Indicates what action the originator requests from the recipient. The following standard values have object identifiers defined in the EDI messaging standards (see Appendix C, "Referenced Documents). The first column of the table lists the symbolic values. The second column specifies in ASN.1 the object identifiers denoted by the values.

Table 7-23 Selected Values of the Action Request Attribute

Value	Object Identifier (ASN.1)
for-action	{joint-iso-ccitt mhs-motis(6) edims(7) 13 0}
copy	{joint-iso-ccitt mhs-motis(6) edims(7) 13 1}

The default value of the attribute is for-action. Additional values for this attribute can be defined by any interested parties.

EDI Notification Requests
Contains requests made of the preferred recipient denoted by the Recipient attribute. Requests may include the type of notification and security constraints on the notification.

Responsibility Passing Allowed
Indicates whether forwarding of Responsibility is allowed. A recipient of a message with the Responsibility Passing Allowed attribute set to FALSE shall originate EDIN's as requested, and shall not forward Responsibility. If this attribute is set to TRUE, Responsibility may be forwarded to at most one recipient.

Interchange Recipient
The Interchange Recipient identifies the EDI Interchange recipient and is semantically identical to the “Interchange recipient” of the EDIFACT UNB segment.

Recipient Reference
Identifies a reference meaningful to the recipient's EDI application. This attribute is semantically identical to the “Recipient's Reference, Password” of the EDIFACT UNB segment.

Interchange Control Reference
Indicates the Interchange Control Reference as assigned by the Interchange sender. This is semantically identical to the “Interchange control reference” of the EDIFACT UNB segment.

Processing Priority Code
Indicates the EDI application Processing Priority Code. This attribute is semantically identical to the “Processing priority code” in the EDIFACT UNB segment. It consists of a string.

Acknowledgement Request
Indicates the request for EDI acknowledgement as indicated by the interchange sender. This attribute is semantically identical to the “Acknowledgement request“ in the EDIFACT UNB segment. Its value is a Boolean, where the value TRUE indicates a request for acknowledgement. Absence of this attribute shall be interpreted as the value FALSE.

Communications Agreement Id
Indicates the type of Communications Agreement controlling the interchange, for example, a Customs agreement. This attribute is semantically identical to the “Communications agreement id” in the EDIFACT UNB segment.

Test Indicator
Indicates that the EDI Interchange is a test. This attribute is semantically identical to the “Test indicator” in the EDIFACT UNB segment. The value TRUE indicates that the EDI Interchange is a test. Absence of this attribute shall be interpreted as the value FALSE.

Authorization Information
Indicates who authorized the interchange. This attribute is semantically identical to the “Authorization information” in the ANSIX12 Interchange.

Recipient Extensions
Contains extensions to the Recipient class.

7.2.36 Security Elements

An instance of class **Security Elements** is used to indicate whether “proof or non-repudiation of content received,” or “EDI application security” services are required for the notification.

An instance of this class has the attributes of its superclass *(Object*) and additionally the attributes listed below.

Table 7-24 Attributes Specific to Security Elements

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
Original Content	Object(Content)	-	0-1	-
Original Content Integrity Check	Object(Integrity Check Basis)	-	0-1	-
EDI Appl Security Elements	Object(EDI Appl Sec Elements)	-	0-1	-

Original Content
Content of the subject EDIM. The Content class is part of the Message Handling Package defined in Chapter 5, “Message Handling Packages”.

Original Content Integrity Check
Provides the notification recipient with a means of validating that the message content of the EDIM has not been modified. The use of this attribute is described in the messaging standards.

EDI Application Security Elements
Identifies a set of security elements. This attribute is used to allow an EDI application to exchange security elements having an end-to-end significance.

7.2.37 Service String Advice

An instance of class **Service String Advice** indicates the service string advice of an EDI Interchange and is semantically identical to the “UNA, Service string advice” of the EDIFACT Interchange.

An instance of this class has the attributes of its superclass (*Object*) and additionally the attributes listed below.

Table 7-25 Attributes Specific to Service String Advice

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
Component Data Element Separator	String(Octet)	1	1	-
Data Element Separator	String(Octet)	1	1	-
Decimal Notation	String(Octet)	1	1	-
Release Indicator	String(Octet)	1	0-1	-
Reserved	String(Octet)	1	0-1	-
Segment Terminator	String(Octet)	1	1	-

Further details on the meaning of this class and its attributes can be found in the EDIFACT standard (see Appendix C, “Referenced Documents”).

7.2.38 Syntax Identifier

An instance of class **Syntax Identifier** indicates the syntax used in an EDI Interchange and is semantically identical to the "Syntax identifier" of the EDIFACT UNB segment.

An instance of this class has the attributes of its superclass (*Object*) and additionally the attributes listed below.

Table 7-26 Attributes Specific to Syntax Identifier

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
Syntax Identifier String	String(Teletex)	1-4	1	-
Syntax Version	String(Printable)	1-5	1	-

Further details on the meaning of this class and its attributes can be found in the EDIFACT standard (see Appendix C, "Referenced Documents).

Message Store Package 8

This chapter, along with Chapter 9, “Message Store Package Errors” defines the Message Store Package. The Object Identifier associated with this package is represented by the constant **MS-Package** *{MS_PACKAGE}*.

8.1 MS Class Hierarchy

This section depicts the hierarchical organization of the MS classes. Subclassification is indicated by indentation. The names of abstract classes are in italics. Thus, for example, **Fetch-Attribute-Defaults** is an immediate subclass of *Attribute-Defaults*, an abstract class. This in turn is an immediate subclass of the abstract class *Object*. The names of classes to which the `OM Encode` function applies are in bold. The `OM Create` function applies to all concrete classes.

The following symbols denote Funtional Units (FUs) in which a class is used, where a class is defined, and classes that a client is not permitted to create or modify:

Symbol	Description
[M]	Class is used in MS FU
[S]	Class is used in MS Submission FU
[A]	Class is used in MS Administration FU
[C]	Class is used in MS Alert FU
{X}	Client is not permitted to create or modify instances of this class because this OM class is only returned by the interface and never supplied to it.
(*)	Class is defined in Chapter 5, “Message Handling Packages”
(**)	Class is defined in Chapter 9, “Message Store Package Errors”

Object (see the *Solstice XOM Programming Reference Manual*)

- Address [A]
- Alert-Address [M]
- Attribute [M]
  - — AVA [M]
  - — Filter-Item [M]
- *Attribute-Defaults* [M]
  - — Fetch-Attribute-Defaults [M]
  - — List-Attribute-Defaults [M]
- Attribute-Selection [M]
- *Auto-Action* [M]
  - — Auto-Action-Deregistration [M]
  - — Auto-Action-Registration [M]
- Auto-Alert-Registration-Parameter [M]
- Auto-Forward-Registration-Parameter [M]
- Bind-Argument [M]
- Bind-Result [M] {X}

- Change-Credentials [M]
- Check-Alert-Result [C] {X}
- *Common-Controls* [A]
  — Default-Delivery-Controls [A]
- Creation-Time-Range [M]
- Credentials [M]
- Delete-Argument [M]
- Deliverable-Content-Types [A]
- EITs [A] (*)
- *Error* [M] (**) {X}
- Fetch-Argument [M]
- Fetch-Result [M] {X}
- Filter [M]
- Items [M]
  — Item [M]
- Label-And-Redirection [A]
- Labels-And-Redirections [A]
- List-Argument [M]
- List-Result [M] {X}
- MS-Entry-Information [M] {X}
- MS-Entry-Information-Selection [M]
- MTS Identifier [S] (*)
- *Name* [M], [S], [A] (*)
  — DS-DN [M], [S], [A] (*)
- OR Name [M], [S], [A] (*)
- Password [M]
- Range [M]
- Register-Argument [A]

- Register-MS-Argument [M]
- *Relative-Name* [M], [S], [A] (*)
  — DS-RDN [M], [S], [A] (*)
- Restrictions [M]
- Security Label [M] (*)
- Selector [M]
- Sequence-Number-Range [M]
- Session [M]
- Strong-Credentials [M]
- *Submitted Communique* [S] (*)
  — Submitted Message [S] (*)
  — Item-To-Forward [S]
  — Auto-Forward-Arguments [S]
  — Submitted Probe [S] (*)
- Submission Results [S] (*) {X}
- Summarize-Argument [M]
- Summary [M] {X}
- Summary-Present [M] {X}
- Summary-Requests [M]
- Summary-Result [M] {X}
- Wait-Result [M] {X}

Note – The terms “Entry-Information” and “Entry-Information-Selection” are used in both X.500 and the MS protocols but they have different ASN.1 syntax definitions. Hence, this MS-specific chapter makes the distinction by using “MS-Entry-Information” and “MS-Entry-Information-Selection”.

8.2 MS Class Definitions

This section defines the classes of the MS package. Error classes are defined in Chapter 9, “Message Store Package Errors”.

8.2.1 Address

The OM class **Address** represents the address of a particular entity or service. An address is an unambiguous name, label or number that identifies the location of the entity or service.

Address is an abstract class that has the OM attributes of its superclass (*Object*) and no other OM attributes.

This class belongs to the Directory Services package, defined in the XDS Specification.

8.2.2 Alert-Address

The OM class **Alert-Address** gives the alert information to used by the auto-alert operation.

An instance of OM class **Alert-Address** has the OM attributes of its superclass *Object* and additionally the OM attributes listed in Table 8-1.

Table 8-1 Attributes Specific to Alert-Address

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
Address	Object(External)	-	1	-
Alert-Qualifier	String(Octet)	-	0-1	-

Address

Identifies the type of alert address to be invoked.

Alert-Qualifier

Contains any further information which need to be included with the auto-alert.

8.2.3 Attribute

An instance of OM class **Attribute** is an attribute of an object and thus a component of its directory entry.

An instance of this OM class has the OM attributes of its superclass *Object* and additionally the OM attributes listed in Table 8-2.

Table 8-2 Attributes Specific to Attribute

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
Attribute-Type	String(Object-Identifier)	-	1	-
Attribute-Values	any	-	0 or more	-

Attribute-Type

The attribute type, which indicates the class of information given by this attribute. See Table 10-1 on page 10-3 and Table 11-1 on page 11-3 for a complete list of MS general and IM attribute types, together with the object identifiers associated with each of them.

Attribute-Values

The attribute values. The OM value syntax and the number of values allowed for this OM attribute are determined by the value of the Attribute-Type OM attribute in accordance with the rules set out in Section 4.2.3.1, “Attribute,” on page 4-5.

Where the values of this OM attribute have syntax String(*), they may be long, segmented strings. For this reason, the functions OM-Read and OM-Write should be used to access all String(*) values.

Note – A directory attribute must always have at least one value, even though instances of this OM class are permitted to have none.

This class belongs to the Directory Services package, defined in the XDS Specification.

8.2.4 Attribute-Defaults

The OM class *Attribute-Defaults* specifies a default set of attribute-types to indicate which attributes should be returned for any subsequent `ms_list` or `ms_fetch` functions if the Entry-Information-Selection argument is absent. If absent, no change will be applied to the registered default.

An instance of this OM class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 8-3.

Table 8-3 Attributes Specific to Attribute-Defaults

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
Attribute-Type	String(Object-Identifier)	-	0/more	-

Attribute-Type

This specifies the default set of attribute-types to indicate which attributes should be returned for any subsequent `ms_list` or `ms_fetch` functions if the Requested-Attributes OM attribute of List-Argument or Fetch-Argument respectively were absent. If absent, no change will be applied to the registered default.

8.2.5 Attribute-Selection

An instance of OM class **Attribute-Selection** describes the selection of attributes of an entry being requested.

An instance of this OM class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 8-4.

Table 8-4 Attributes Specific to Attribute-Selection

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
Attribute-Type	String(Object-Identifier)	-	1	-
From	Integer	-	0-1	-
Count	Integer	-	0-1	-

Attribute-Type

This indicates the class of information given by this MS attribute.

From

This may only be present if the attribute-type is multi-valued. When an MS attribute is multi-valued, this indicates the relative position of the first value to be returned. If it specifies a value beyond those present in the MS attribute, no values are returned. If it is omitted, values starting at the first value are returned.

Note – MS attributes are numbered starting at position 1.

Count

This may only be present if the attribute-type is multi-valued. When an MS attribute is multi-valued, this gives the number of values to be returned. If there are less than count values present in the MS attribute, all values are returned. If it is omitted, there is no limit as to how many values are returned.

8.2.6 AVA

An instance of OM class **AVA** (Attribute Value Assertion) is a proposition concerning the values of a directory entry.

An instance of this OM class has the OM attributes of its superclasses *Object*, **Attribute** and no other OM attributes. There is an additional constraint on this OM class, in that there must be exactly one value of the OM attribute **Attribute-Values**. The **Attribute-Type** remains single-valued. The OM value syntax of **Attribute-Values** must conform to the rules set out in Section 4.2.3.2, “AVA,” on page 4-6.

This class belongs to the Directory Services package, defined in the XDS Specification.

8.2.7 Auto-Action

The OM class **Auto-Action** describes an action that will occur automatically whenever the associated registration criteria have been satisfied. The result of an action being invoked is visible externally to the MS. Auto-actions are registered with the MS using the `ms_register_ms` function.

This OM class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 8-5.

Table 8-5 Attributes Specific to Auto-Action

OM Attribute	**Value Syntax**	**Value Length**	**Value Number**	**Value Initially**
Type	String(Object-Identifier)	-	1	-
Registration-ID	Integer	-	1	-

Type

This indicates the class of auto-action type.

Registration-ID

This identifies a particular auto-action registration.

8.2.8 Auto-Action-Deregistration

An instance of OM class **Auto-Action-Deregistration** describes an auto-action-registration to be deregistered using the `ms_register_ms` function. Any auto-action with registration-identifier and auto-action-type matching that specified is deregistered.

An instance of this OM class has the OM attributes of its superclasses *Object*, **Auto-Action** and no additional OM attributes.

8.2.9 Auto-Action-Registration

An instance of OM class **Auto-Action-Registration** describes an auto-action-registration to be registered using the `ms_register_ms` function. The new auto-action-registration-parameter supersedes any previously registered auto-action (if any) with that registration-identifier and auto-action-type.

An instance of this OM class has the OM attributes of its superclasses *Object*, **Auto-Action** and additionally, the OM attributes listed in Table 8-6.

Table 8-6 Attributes Specific to Auto-Action-Registration

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
Registration-Parameter	Any	-	1	-

Registration-Parameter

This specifies the object identifier of an auto-action type to which an auto-action registration must conform.

8.2.10 Auto-Alert-Registration-Parameter

An instance of OM class **Auto-Alert-Registration-Parameter** specifies the criteria to determine whether a user should be alerted to the delivery of a message into the stored message information base.

An instance of this OM class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 8-7.

Table 8-7 Attributes Specific to Auto-Alert-Registration-Parameter

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
Filter	Object(Filter)	-	0-1	-
Alert-Address	Object(Alert-Address)	-	1/more	-
Requested-Attributes	Object(MS EntryInformationSelection)	-	0-1	-

Filter

The set of criteria which a delivered message must satisfy for the auto alert function to be activated with the given set of parameters.

Alert-Address

The types of alert services to be invoked.

Requested-Attributes

This indicates what information from the selected entries is to be included with the auto-alert.

8.2.11 Auto-Forward-Arguments

An instance of OM class **Auto-Forward-Arguments** describes the set of criteria to be used in the auto-forwarding of a submitted message.

An instance of this OM class has the OM attributes of its superclass *Object*, and additionally the attributes listed in Table 8-8.

Table 8-8 Attributes Specific to Auto-Forward-Arguments

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
Alternate-Recipient-Allowed	Boolean	-	1	false
Confidentiality-Algorithm	Object (Algorithm)	-	0-1	-
Content-Correlator	*any*	3-512	0-1	-
Content-Identifier	String (Printable)	1-16	0-1	-
Content-Return-Requested	Boolean	-	1	false
Conversion-Loss-Prohibited	Boolean	-	1	false
Conversion-Prohibited	Boolean	-	1	false
Deferred-Delivery-Time	String (UTC Time)	0-17	0-1	-
Disclosure-Allowed	Boolean	-	1	false
Expansion-Prohibited	Boolean	-	1	false
Latest-Delivery-Time	String (UTC Time)	0-17	0-1	-
Origin-Check	Object (Algorithm and Result)	-	0-1	-
Original-EITs	Object (EITs)	-	0-1	-

Table 8-8 Attributes Specific to Auto-Forward-Arguments *(Continued)*

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
Originator-Certificate	Object (Certificates)[1]	-	0-1	-
Originator-Name	Object (OR Name)	-	1	-
Originator-Return-Address	Object (OR Address)	-	0-1	-
Priority	Enum (Priority)	-	1	normal
Proof-of-Submission-Requested	Boolean	-	1	false
Reassignment-Prohibited	Boolean	-	1	false
Recipient-Descriptors	Object (RD)	-	1-32767	-
Security-Label	Object (Security Label)	-	0-1	-

[1]As defined in the XDS Specification (see Appendix C, “Referenced Documents”).

Alternate-Recipient-Allowed

Whether the originator permits the MTS to deliver the subject message to an alternate recipient. A management domain (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.

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 recipient descriptors 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-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-Return-Requested

Whether the Content attribute is to be included as the like-named attribute of any NDRs the submitted message provokes.

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.

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.

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.

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.

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.

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 10.3.3, "Priority," on page 10-9.

Proof-of-Submission-Requested

Whether the originator of the submitted message requires proof of its submission.

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 must be assigned in line with the security policy in force.

8.2.12 Auto-Forward-Registration-Parameter

An instance of OM class **Auto-Forward-Registration-Parameter** specifies the criteria to determine whether a delivered message should be forwarded.

An instance of this OM class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 8-9.

Table 8-9 Attributes Specific to Auto-Forward-Registration-Parameter

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
Auto-Forward-Arguments	Object(Auto-Forward-Arguments)	-	1	-
Delete-After-Auto-Forward	Boolean	-	1	true
Filter	Object(Filter)	-	0-1	-
Other-Parameters	String(Octet)	-	0-1	-

Auto-Forward-Arguments

The set of arguments to be used for each auto-forward message-submission operation.

Delete-After-Auto-Forward

If true, an entry is to be deleted after auto-forwarding. If false, it is not deleted.

Filter

The set of criteria which a delivered message must satisfy for the message to be auto-forwarded with the given set of parameters.

Other-Parameters

Optional extra information to be used in auto-forwarding.

8.2.13 Bind-Argument

An instance of OM class **Bind-Argument** specifies information necessary for establishing a session with the Message Store, together with details of the service required.

An instance of this OM class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 8-10.

Table 8-10 Attributes Specific to Bind-Argument

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
Initiator	Object(OR-Name) [1]	-	1	-
Initiator-Credentials	Object(Credentials)	-	1	-
Security-Context	Object(Security-Label)	-	0/more	-
Fetch-Restrictions	Object(Restrictions)	-	0-1	-
MS-Configuration-Request	Boolean	-	1	false

[1]As defined in Section 5.2.38, “OR Name,” on page 5-58.

Initiator

This specifies the OR-Name of the initiator (the UA) of this session (or association) with the MS.

Initiator-Credentials

This specifies the credentials of the initiator for authentication purposes.

Security-Context

This identifies the security context at which the initiator proposes to operate.

Fetch-Restrictions

This specifies the restrictions on entries to be returned as result of a `ms_fetch` function. These restrictions prevail until the `ms_unbind` function is issued.

MS-Configuration-Request

If true, this specifies the request to obtain information relating to which auto-actions and optional attributes the MS provides support for. If false, no such request is being made.

8.2.14 Bind-Result

An instance of OM class **Bind-Result** describes the result returned from the `ms_bind` function. It provides information about the MS capabilities, if requested.

An instance of this OM class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 8-11.

Table 8-11 Attributes Specific to Bind-Result

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
Responder-Credentials	Object(Credentials)	-	1	-
Available-Auto-Actions	String(Object-Identifier)	-	0/more	-
Available-Attribute-Types	String(Object-Identifier)	-	0/more	-
Alert-Indication	Boolean	-	1	false
Content-Types-Supported	String(Object-Identifier)	-	0/more	-

Responder-Credentials

This contains the credentials of the MS.

Available-Auto-Actions

This specifies the set of all possible auto-actions that are supported by the MS (not just those requested by the UA). This is only present if the MS-Configuration-Request was made for the `ms_bind` function.

Available-Attribute-Types

This specifies the set of all optional MS attribute-types that are supported by the MS. This is only present if the MS-Configuration-Request was made for the `ms_bind` function.

Alert-Indication

If true, this indicates an alert condition has occurred since the last successful Alert-indication.

Content-Types-Supported

This specifies a set of object-identifiers defining the content-types of which the MS has knowledge. This is only present if the MS-Configuration-Request was made for the `ms_bind` function.

8.2.15 Change-Credentials

An instance of OM class **Change-Credentials** gives the user's current (old) credentials and the new credentials to which the user would like to change.

An instance of this OM class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 8-12.

Table 8-12 Attributes Specific to Change-Credentials

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
Old-Credentials	Object(Credentials)	-	1	-
New-Credentials	Object(Credentials)	-	1	-

Old-Credentials

This specifies the user's current credentials. This may be credentials for either simple or strong authentication.

New-Credentials

This specifies the credentials to which the user would like to change. This shall be the same type (simple or strong) as the **Old-Credentials**.

8.2.16 Check-Alert-Result

An instance of OM class **Check-Alert-Result** gives information regarding an alert when a new entry has been entered into the Message Store in response to the `ms_check_alert` function.

An instance of this OM class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 8-13.

Table 8-13 Attributes Specific to Check-Alert-Result

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
Alert-Registration-Identifier	Integer	-	1	-
New-Entry	Object(MS-Entry-Information)	-	0-1	-

Alert-Registration-Identifier

This identifies which auto alert registrations resulted in the alert.

New-Entry

If present, this conveys the information from the new entry which was requested in the auto alert registration parameter. This is absent if the user did not specify an auto alert registration parameter.

8.2.17 Common-Controls

The OM class **Common-Controls** indicates the control parameters, which are related to delivery and submission, common to the classes Default-Delivery-Controls, Delivery-Controls and Submission-Controls (although the latter two are not used by functions defined in this specification).

An instance of this OM class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 8-14.

Table 8-14 Attributes Specific to Common-Controls

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
Restrict	Boolean	-	1	true
Permissible-Operations	String(Bit)	-	0-1	-
Permissible-Maximum-Content-Length	Integer	-	0-1	-
Permissible-Lowest-Priority	Enum(Priority)[1]	-	0-1	-

[1]As defined in Section 5.3.9, “Priority,” on page 5-90

Note – The following control parameters apply either to delivery or to submission (the latter being indicated in brackets []). For further details, see X.411.

Restrict

This indicates whether the controls on the delivery-port [or submission-port] abstract operations are to be updated ('true') or removed ('false').

Permissible-Operations

This indicates the abstract operations that the MTS may invoke on the MS or UA.

Permissible-Maximum-Content-Length

This indicates the content-length, in octets, of the longest content message that the MTS shall deliver to the MS or UA via the Message-Delivery abstract operation [or the MS or UA shall submit to the MTS via the Message-Submission abstract operation].

Permissible-Lowest-Priority

This indicates the priority of the lowest priority message that the MTS shall deliver to the MS or UA via the Message-Delivery abstract operation [or the MS or UA shall submit to the MTS via the Message-Submission abstract operation].

8.2.18 Creation-Time-Range

An instance of OM class **Creation-Time-Range** identifies a contiguous sequence of entries based on their times of creation.

An instance of this OM class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 8-15.

Table 8-15 Attributes Specific to Creation-Time-Range

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
From	String(UTC-Time)	-	0-1	-
To	String(UTC-Time)	-	0-1	-

From

This indicates creation-time that is the lower bound for the range. If absent, the default is no lower bound; and the selection begins with the entry with the earliest creation-time in the information base.

To

This indicates creation-time that is the upper bound for the range. If absent, the default is no upper bound; and the selection finishes with the entry with the latest creation-time in the information base.

8.2.19 Credentials

An instance of OM class **Credentials** holds either a simple password when simple authentication is used or strong credentials when strong authentication is used.

An instance of this OM class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 8-16.

Table 8-16 Attributes Specific to Credentials

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
Simple	Object(Password)	-	0-1	-
Strong	Object(Strong-Credentials)	-	0-1	-

Simple

This is the password used for simple authentication.

Strong

This specifies the credentials used for strong authentication.

Note – No instance will contain both of the above OM attributes.

8.2.20 Default-Delivery-Controls

An instance of OM class **Default-Delivery-Controls** gives the default control parameters related to delivery.

An instance of this OM class has the OM attributes of its superclasses *Object*, **Common-Controls** and additionally, the OM attributes listed in Table 8-17.

Table 8-17 Attributes Specific to Default-Delivery-Controls

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
Permissible-Content-Types-Int	Integer[1]	-	0/more	-
Permissible-Content-Types	String(Object-Identifier)[1]	-	0/more	-
Permissible-EITs	Object(EITs)[2]	-	0-1	-

[1] For services based on MHS-1984, only attribute “Permissible-Content-Types-Int” is available. For services based on MHS-1988, either “Permissible-Content-Types-Int” or “Permissible-Content-Types” is available.

[2] As defined in Section 8.2.25, “EITs,” on page 8-24.

Permissible-Content-Types-Int

Permissible-Content-Types

These indicate the content-types that shall appear in messages the MTS shall deliver to the MS via the Message-Delivery abstract operation. **Content-Type** identifies the syntax and semantics of the value of the OM attribute Content of the OM class Message. Its defined values are as prescribed for the like-named attribute specific to the OM class Communique of the Message Handling Package (see Section 5.2.7, “Communique,” on page 5-8).

Permissible-EITs

This indicates the encoded-information-types that shall appear in messages the MTS shall deliver to the MS via the Message-Delivery abstract operation.

8.2.21 Delete-Argument

An instance of OM class **Delete-Argument** describes the arguments for the `ms_delete` function.

An instance of this OM class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 8-18.

Table 8-18 Attributes Specific to Delete-Argument

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
Information-Base-Type	Integer	-	0-1	stored-messages
Items	Object(Items)	-	1	-

Information-Base-Type

Specifies the repository containing entries for delivered messages and reports. Its value is **stored-messages** *{MS_STORED_MESSAGES}*.

Items

This determines which entries are to be deleted.

8.2.22 Deliverable-Content-Types

An instance of OM class **Deliverable-Content-Types** indicates which content-types that the MTS shall permit to appear in messages delivered to the MS.

An instance of this OM class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 8-19.

Table 8-19 Attributes Specific to Deliverable-Content-Types

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
Content-Type-Int	Integer	-	1/more	-
Content-Type	String(Object-Identifier)	-	1/more	-

Content-Type-Int

Content-Type

These identify the syntax and semantics of the value of the OM attribute Content of the OM class Message. The Content Type defined values are as prescribed for the like-named attribute specific to the OM class Communique of the MH Package (see Section 5.2.7, “Communique,” on page 5-8).

Note – For services based on MHS-1984, only attribute Content-Type-Int is available. For services based on MHS-1988, either Content-Type-Int or Content-Type is available.

8.2.23 DS-DN

As defined in Section 5.2.17, “DS-DN,” on page 5-26.

8.2.24 DS-RDN

As defined in Section 5.2.18, “DS-RDN,” on page 5-27.

8.2.25 EITs

As defined in the Message Handling Package in Section 5.2.19, “EITs,” on page 5-27.

8.2.26 Error

See Chapter 9, “Message Store Package Errors”.

8.2.27 Fetch-Argument

An instance of OM class **Fetch-Argument** describes the arguments for the `ms_fetch` function.

An instance of this OM class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 8-20.

Table 8-20 Attributes Specific to Fetch-Argument

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
Information-Base-Type	Integer	-	0-1	stored-messages
Item	Object(Item)	-	1	-
Requested-Attributes	Object(MS-Entry-Information-Selection)	-	0-1	-

Information-Base-Type

Specifies the repository containing entries for delivered messages and reports. Its value is **stored-messages** *{MS_STORED_MESSAGES}*.

Item

This determines which entry is to be fetched.

Requested-Attributes

This indicates what information from the selected entry is to be returned in the result. If absent, it implies that information about the entry itself, rather than the attributes of the entry, is requested.

8.2.28 Fetch-Attribute-Defaults

An instance of OM class **Fetch-Attribute-Defaults**, for the `ms_register_ms` function, specifies a default set of attribute-types to indicate which attributes should be returned for any subsequent `ms_fetch` functions if the Entry-Information-Selection argument is absent. This value replaces any previously registered default set.

An instance of this OM class has the OM attributes of its superclasses *Object*, **Attribute-Defaults**, and no additional OM attributes.

8.2.29 Fetch-Result

An instance of OM class **Fetch-Result** gives the result of a successful `ms_fetch` function.

An instance of this OM class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 8-21.

Table 8-21 Attributes Specific to Fetch-Result

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
Entry-Information	Object(MS-Entry-Information)	-	0-1	-
List	Integer	-	0/more	-
Next	Integer	-	0-1	-

Entry-Information

This is the set of MS attributes from one entry as requested in the argument to the `ms_fetch` function. This is not present if a search was performed but no entry was selected.

List

This is returned in the case that a search was performed and more than one entry was found that matched the search selector. This would then give the sequence-numbers, in ascending order, of these further entries.

Next

This is returned in the case where the number of entries selected would have been greater if it were not for the limit specified in the selector. This would then give the sequence-number for the next entry that would have been selected.

8.2.30 Filter

An instance of OM class **Filter** is a basis for selecting or rejecting an object on the basis of information in its directory entry. At any point in time, an attribute filter has a value relative to every object. The value is true or false. The object is selected if, and only if, the filter’s value is true.

A filter is a collection of simpler filters and elementary filter-items together with a Boolean operation. The filter value is undefined if and only if all the component Filters and Filter-Items are undefined. Otherwise, the filter has a Boolean value with respect to any directory entry, which can be determined by evaluating each of the nested components and combining their values using the Boolean operation.

An instance of this OM class has the OM attributes of its superclass *Object* and additionally the OM attributes listed in Table 8-22.

Table 8-22 Attributes Specific to Filter

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
Filter-Items	Object(Filter-Item)	-	0 or more	-
Filters	Object(Filter)	-	0 or more	-
Filter-Type	Enum(Filter-Type)	-	1	and

Filter-Items

A collection of assertions, each relating to just one attribute of a directory entry.

Filters

A collection of simpler filters.

Filter-Type

The filter's type. Its value may be:

- **and**, meaning that the filter is the logical conjunction of its components. The filter is true unless any of the nested filters or filter items is false. If there are no nested components, the filter is true.
- **or**, meaning that the filter is the logical disjunction of its components. The filter is false unless any of the nested filters or filter items is true. If there are no nested components, the filter is false.
- **not**, meaning that the result of this filter is reversed. There must be exactly one nested filter or filter item. The filter is true if the enclosed filter or filter item is false, and is false if the enclosed filter or filter item is true.

This class belongs to the Directory Services package, defined in the XDS Specification.

8.2.31 Filter-Item

An instance of OM class **Filter-Item** is a component of a **Filter**. It is an assertion about the existence or values of a single attribute type in a directory entry.

An instance of this OM class has the OM attributes of its superclasses *Object*, **Attribute** and additionally the OM attributes listed in Table 8-23 below.

Table 8-23 Attributes Specific to Filter-Item

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
Filter-Item-Type	Enum(Filter-Item-Type)	-	1	-
Final-Substring	String(*)	1 or more	0-1	-
Initial-Substring	String(*)	1 or more	0-1	-

Note – The OM attributes **Attribute-Type** and **Attribute-Values** are inherited from the superclass **Attribute**.

This class belongs to the Directory Services package, defined in the XDS Specification. There are some differences regarding the interpretation of filters as defined by the Directory and the MS standards:

- In the MS, a Filter-Item may only be **true** or **false**; in the Directory it may be **true**, **false**, or undefined.
- In the MS, the definitions of the **greater-or-equal** and **less-or-equal** Filter-Item-Types are the opposite of those in the Directory. Also, they do not result in **true** in the event of equality.

The value of the filter item is undefined if any of the following conditions apply:

- the **Attribute-Type** is not known
- any of the **Attribute-Values** do not conform to the attribute syntax defined for that attribute type
- the **Filter-Item-Type** uses a matching rule that is not defined for the attribute syntax

Access control restrictions may also cause the value to be undefined.

Filter-Item-Type

Identifies the type of the filter item and thereby the nature of the filter. Its value must be one of the following:

- **approximate-match**, meaning that the filter is true if the directory entry contains at least one value of the specified type that is approximately equal to that specified (the meaning of “approximately equal” being implementation-dependent), and false otherwise.

  Rules for approximate matching are locally defined. For example, an approximate match might take into account spelling variations or employ phonetic comparison rules. In the absence of any such capabilities, a DSA should treat an approximate match as a test for equality.

  There must be exactly one value of the OM attribute **Attribute-Values**.

- **equality**, meaning that the filter is false.

  There must be exactly one value of the OM attribute **Attribute-Values**.

- **greater-or-equal**, meaning that the filter item is true if, and only if, the supplied value is greater than or equal to at least one value of the attribute (using the appropriate ordering algorithm).

  There must be exactly one value of the OM attribute **Attribute-Values**.

- **less-or-equal**, meaning that the filter item is true if and only if the supplied value is less than or equal to at least one value of the attribute (using the appropriate ordering algorithm).

  There must be exactly one value of the OM attribute **Attribute-Values**.

- **present**, meaning that the filter is true if the entry contains an attribute of the specified type, and false otherwise.

  Any values of the OM attribute **Attribute-Values** are ignored.

- **substrings**, meaning that the filter is true if the entry contains at least one value of the specified attribute type that contains all of the specified substrings in the given order, and false otherwise.

  There can be any number of substrings given as values of the OM attribute **Attribute-Values**, including none. There can also be a substring in **Initial** and/or **Final**. The substrings shall be non-overlapping, but they may be separated from each other or from the ends of the attribute value by zero or more string elements. However, there must exist at least one attribute of type Attribute-Values, Initial-Substring, or Final-Substring.

Final-Substring

If present, the substring that is to match the final portion of an attribute value in the entry. This attribute may only exist if the Filter-Item-Type is equal to ‘substrings’.

Initial-Substring

If present, the substring that is to match the initial portion of an attribute value in the entry. This attribute may only exist if the Filter-Item-Type is equal to ‘substrings’.

8.2.32 Item

An instance of the OM class **Item** identifies the criterion for selecting a single MS entry to be returned by the `ms_fetch` function.

An instance of this OM class has the OM attributes of its superclasses *Object*, **Items** and no additional OM attributes.

Additional Constraints

- **Selector**
  If present, this specifies a set of entries of which the one with the lowest sequence-number is the entry to be chosen.
- **Precise**
  If present, there shall be precisely a single value which would identify an MS entry by its sequence-number.

8.2.33 Item-To-Forward

An instance of the OM class **Item-To-Forward** gives the sequence-number identifying the single entry that is to be forwarded by the `ms_submit` function.

Note – The entry to be forwarded should be a delivered message entry. Forwarding of entries that are not delivered messages is not defined in this book.

An instance of this OM class has the OM attributes of its superclasses *Object*, **Submitted-Message** and the additional OM attributes listed in Table 8-24.

Table 8-24 Attributes Specific to Item-To-Forward

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
Sequence-Number	Integer	-	1	-

Sequence-Number

This identifies the MS entry, by its sequence number, that is to be forwarded via indirect submission.

8.2.34 Items

An instance of the OM class **Items** identifies the criterion for selecting MS entries to be removed by the `ms_delete` function.

An instance of this OM class has the OM attributes of its superclass *Object* and additionally, either of the OM attributes listed in Table 8-25.

Table 8-25 Attributes Specific to Items

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
Selector	Object(Selector)	-	0-1	-
Precise	Integer	-	0/more	-

Selector

If present, this specifies a set of entries to be chosen.

Precise

If present, this is a list of sequence-number(s) to precisely identify the entries to be chosen based on their sequence-numbers.

Note – No instance can contain both of the above OM attributes.

8.2.35 Label-And-Redirection

An instance of OM class **Label-And-Redirection** indicates a set of security-label and the OR-Name of an alternate-recipient.

An instance of this OM class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 8-26.

Table 8-26 Attributes Specific to Label-And-Redirection

OM Attribute	**Value Syntax**	**Value Length**	**Value Number**	**Value Initially**
Security-Label	Object(Security-Label)[1]	-	0-1	-
Alternate-Recipient	Object(OR-Name)[2]	-	0-1	-

[1]As defined in Section 5.2.48, “Security Label,” on page 5-67.
[2]As defined in Section 5.2.38, “OR Name,” on page 5-58.

Security-Label

This identifies the user-security-label.

Alternate-Recipient

This identifies the OR-Name of an alternate-recipient.

8.2.36 Labels-And-Redirections

An instance of OM class **Labels-And-Redirections** indicates a set of security-label and the OR-Name of alternate-recipient pair.

An instance of this OM class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 8-27.

Table 8-27 Attributes Specific to Labels-And-Redirections

OM Attribute	**Value Syntax**	**Value Length**	**Value Number**	**Value Initially**
Label-And-Redirection	Object(Label-And-Redirection)	-	1/more	-

Label-And-Redirection

This identifies either one or both of the following:

- a security-label
- the OR-Name of an alternate-recipient

8.2.37 List-Argument

An instance of OM class **List-Argument** describes the arguments for the `ms_list` function.

An instance of this OM class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 8-28.

Table 8-28 Attributes Specific to List-Argument

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
Information-Base-Type	Integer	-	0-1	stored-messages
Selector	Object(Selector)	-	1	-
Requested-Attributes	Object(MS-Entry-Information-Selection)	-	0-1	-

Information-Base-Type

Specifies the repository containing entries for delivered messages and reports. Its value is **stored-messages** *{MS_STORED_MESSAGES}*.

Selector

This determines which entries are to be listed.

Requested-Attributes

This indicates what information from the selected entry is to be returned in the result.

Note – If absent, it implies that information about the entry itself, rather than the attributes of the entry, is requested.

8.2.38 List-Attribute-Defaults

An instance of OM class **List-Attribute-Defaults**, for the `ms_register_ms` function, specifies a default set of attribute-types to indicate which attributes should be returned for any subsequent `ms_list` function if the Entry-Information-Selection argument is absent. This value replaces any previously registered default set.

An instance of this OM class has the OM attributes of its superclasses *Object*, **Attribute-Defaults**, and no additional OM attributes.

8.2.39 List-Result

An instance of OM class **List-Result** gives the result of a successful `ms_list` function.

An instance of this OM class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 8-29.

Table 8-29 Attributes Specific to List-Result

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
Next	Integer	-	0-1	-
Requested	Object(MS-Entry-Information)	-	0/more	-

Next

This is returned in the case where the number of entries selected would have been greater if it were not for the limit specified in the selector. This would then give the sequence-number for the next entry that would have been selected.

Requested

This is the set of MS attributes of the entries requested in the argument to the `ms_list` function. This is not present if a search was performed but no entries were selected.

8.2.40 MS-Entry-Information

An instance of OM class **MS-Entry-Information** describes information about an MS entry.

An instance of this OM class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 8-30.

Table 8-30 Attributes Specific to MS-Entry-Information

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
Sequence-Number	Integer	-	1	-
Attributes	Object(Attribute)	-	0/more	-

Sequence-Number

This identifies precisely the entry that is to be chosen.

Attributes

This identifies the MS attributes, or those requested, of an MS entry.

8.2.41 MS-Entry-Information-Selection

An instance of OM class **MS-Entry-Information-Selection** indicates which MS attributes of an entry are being requested.

An instance of this OM class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 8-31.

Table 8-31 Attributes Specific to MS-Entry-Information-Selection

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
Selection	Object(Attribute-Selection)	-	1/more	-

Selection

This indicates which MS attributes of an entry are being requested.

8.2.42 MTS Identifier

As defined in the Message Handling Package in Section 5.2.35, “MTS Identifier,” on page 5-46.

8.2.43 Name

As defined in Section 5.2.36, “Name,” on page 5-47.

8.2.44 OR Name

As defined in the Message Handling Package in Section 5.2.38, “OR Name,” on page 5-58.

8.2.45 Password

An instance of OM class **Password** gives the simple password for simple authentication.

An instance of this OM class has the OM attributes of its superclass *Object* and additionally, either of the OM attributes listed in Table 8-32.

Table 8-32 Attributes Specific to Password

OM Attribute	**Value Syntax**	**Value Length**	**Value Number**	**Value Initially**
IA5-String	String(IA5)	-	0-1	-
Octet-String	String(Octet)	-	0-1	-

IA5-String

This specifies the simple password in IA5 string format.

Octet-String

This specifies the simple password in Octet string format.

Note – No instance can contain both of the above OM attributes.

8.2.46 Range

An instance of OM class **Range** is used to select a contiguous sequence of entries, based on either their sequence-numbers or creation-times, from an information base (see Section 8.1.1 in reference X.413, Appendix C, “Referenced Documents”).

An instance of this OM class has the OM attributes of its superclass *Object* and additionally, either of the OM attributes listed in Table 8-33.

Table 8-33 Attributes Specific to Range

OM Attribute	**Value Syntax**	**Value Length**	**Value Number**	**Value Initially**
Sequence-Number-Range	Object(Sequence-Number-Range)	-	0-1	-
Creation-Time-Range	Object(Creation-Time-Range)	-	0-1	-

Sequence-Number-Range

This identifies a contiguous sequence of entries based on their sequence-numbers.

Creation-Time-Range

This identifies a contiguous sequence of entries based on the times the entries were created.

Note – No instance can contain both of the above OM attributes.

8.2.47 Register-Argument

An instance of OM class **Register-Argument** specifies how to modify (using the `ms_register` function) various parameters held by the MTS regarding delivery of messages to the MS.

An instance of this OM class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 8-34.

Table 8-34 Attributes Specific to Register-Argument

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
Name	Object(OR-Name) [1]	-	0-1	-
Address	Object(Address)	-	0-1	-
Deliverable-EIT	Object(EITs) [2]	-	0-1	-
Deliverable-Maximum-Content-Length	Integer	-	0-1	-
Default-Delivery-Controls	Object(Default-Delivery-Controls)	-	0-1	-
Deliverable-Content-Types	Object(Deliverable-Content-Types)	-	0-1	-
Labels-And-Redirections	Object(Labels-And-Redirections)	-	0-1	-

[1]As defined in Section 8.2.44, “OR Name,” on page 8-36.

[2]As defined in Section 8.2.25, “EITs,” on page 8-24.

Name

This specifies the OR-Name of the MS (this corresponds to the OR-Name of the UA), if the user-name is to be changed.

Address

This specifies the address of the MS, if it is required by the MTS and if it is to be changed. This may contain one of the following forms of address:

- the X.121-address and / or the TSAP-ID (transport service access point identifier).
- the PSAP-ID (presentation service access point identifier).

Deliverable-Encoded-Information-Types

This indicates the encoded-information-types that the MTS shall permit to appear in messages delivered to the MS, if they are to be changed. The MTS shall reject as undeliverable any message for an MS for which the MS is not registered to accept delivery of all of the encoded-information-types of the message. Note that the MS may register to receive the **undefined** encoded-information-type. This OM attribute, **Deliverable-Encoded-Information-Types** also indicates the possible encoded-information-types to which implicit conversion can be performed.

Deliverable-Maximum-Content-Length

This indicates the content-length, in octets, of the longest content message that the MTS shall permit to appear in messages being delivered to the MS, if it is to be changed. The MTS shall reject as undeliverable any message for an MS for which the MS is not registered to accept delivery of messages of its size.

Default-Delivery-Controls

This indicates default delivery controls which are registered using the `ms_register` function. The default delivery control OM attributes shall not admit messages whose delivery are prohibited by the prevailing registered values of the **Deliverable-Encoded-Information-Types** OM attribute, the **Deliverable-Content-Types** OM attribute or the **Deliverable-Maximum-Content-Length** OM attribute.

Deliverable-Content-Types

This indicates the content-types that the MTS shall permit to appear in messages delivered to the MS, if they are to be changed. The MTS shall reject as undeliverable any message for an MS for which the MS is not registered to accept delivery of all of the content-types of the message. Note that the MS may register to receive the undefined content-type.

Labels-And-Redirections

This contains either one or both of the following:

- The OR-Name of an alternate recipient to which messages are to be redirected, if this is to be changed.
- The security-label of the UA, if they are to be changed.

8.2.48 Register-MS-Argument

An instance of OM class **Register-MS-Argument** specifies the information to be registered or de-registered through the `ms_register_ms` function.

An instance of this OM class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 8-35.

Table 8-35 Attributes Specific to Register-MS-Argument

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
Auto-Action-Registrations	Object(Auto-Action-Registration)	-	0/more	-
Auto-Action-Deregistrations	Object(Auto-Action-Deregistration)	-	0/more	-
List-Attribute-Defaults	Object(List-Attribute-Defaults)	-	0-1	-
Fetch-Attribute-Defaults	Object(Fetch-Attribute-Defaults)	-	0-1	-
Change-Credentials	Object(Change-Credentials)	-	0-1	-
User-Security-Labels	Object(Security-Label)	-	0/more	-

Auto-Action-Registrations

This is a set of auto-action-registrations, one for each auto-action to be registered.

Auto-Action-Deregistrations

This is a set of auto-action-deregistrations, one for each auto-action to be deregistered.

List-Attribute-Defaults

This specifies a default set of attribute types to indicate which attributes should be returned for any subsequent `ms_list` function if the Requested-Attributes OM attribute of List-Argument is absent.

Fetch-Attribute-Defaults

This specifies a default set of attribute types to indicate which attributes should be returned for any subsequent `ms_fetch` function if the Requested-Attributes OM attribute of Fetch-Argument is absent.

Change-Credentials

If change credentials is requested, this specifies the old and new credentials of the end user.

User-Security-Labels

This contains the security-labels of the UA, if they are to be changed.

8.2.49 Relative Name

As defined in Section 5.2.46, "Relative-Name," on page 5-65.

8.2.50 Restrictions

An instance of OM class **Restrictions** describes the restrictions on entries to be returned as result of a `ms_fetch` function.

An instance of this OM class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 8-36.

Table 8-36 Attributes Specific to Restrictions

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
Allowed-Content-Types	String(Object-Identifier)	-	0/more	-
Allowed-EITs	String(Object-Identifier)	-	0/more	-
Max-Content-Length	Integer	-	0-1	-

Allowed-Content-Types

This specifies the content-types that the UA is prepared to accept as a result of a `ms_fetch` function invocation. Any message with a content-type other than the ones specified will not be returned, but will result in an error, unless the `ms_fetch` function has explicitly overridden the restriction. If absent, the default is that no fetch-restrictions on content-types need to be performed.

Allowed-EITs

This specifies the encoded-information-types that the UA is prepared to accept as a result of a `ms_fetch` function. For any message with an EIT other than the ones specified, a filtering will occur so that disallowed EIT parts are not returned along with the text of the message. If the entire message consists of disallowed EITs, an error will be reported. No filtering will occur if the `ms_fetch` function has been explicitly overridden. If absent, the default is that no fetch-restrictions on EITs need to be performed.

Max-Content-Length

This specifies the maximum content length that the UA is prepared to accept as a result of a `ms_fetch` function invocation. Any message with a content-length exceeding this maximum will not be returned but will result in an error, unless the `ms_fetch` function has explicitly overridden the restriction. If absent, the default is that no fetch-restrictions on content-length need to be performed.

8.2.51 Security Label

As defined in the Message Handling Package in Section 5.2.48, "Security Label," on page 5-67.

8.2.52 Selector

An instance of OM class **Selector** is used to describe the criteria for selecting entries from an information base. The selection operates in three stages, as follows:

1. The total set of entries in the information base may be restricted to a particular contiguous set by specifying its range.
2. Entries from within this set may be selected by specifying a filter which the selected entry must satisfy.
3. A limit may be imposed on the number of entries thus selected; in this case, it is those entries with the lowest sequence-numbers which are selected.

An instance of this OM class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 8-37.

Table 8-37 Attributes Specific to Selector

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
Child-Entries	Boolean	-	1	false
Range	Object(Range)	-	0-1	-
Filter	Object(Filter)	-	0-1	-
Limit	Integer	-	0-1	-
Override	String(Bit)	-	0-1	-

Child-Entries

This indicates whether or not child entries are considered for selection. If **false**, only the main-entries are considered for selection. If **true**, both main-entries and child-entries are considered for selection.

Range

This specifies the contiguous set of entries selected in the first stage of the selection process as mentioned above. If absent, the default is *unbounded*.

Filter

This specifies the filter (or selection criterion) for the second stage of the selection process as mentioned above. If absent, the default is *all entries within the specified range*.

Limit

This is the upper limit on the number of entries selected in the third stage of the selection process as mentioned above. If absent, the default is all entries are returned.

Override

This identifies any override restrictions, if required. See Section 8.1.3 in reference X.413, Appendix C, "Referenced Documents".

8.2.53 Sequence-Number-Range

An instance of OM class **Sequence-Number-Range** identifies a contiguous sequence of entries based their sequence-numbers.

An instance of this OM class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 8-38.

Table 8-38 Attributes Specific to Sequence-Number-Range

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
From-Int	Integer	-	0-1	-
To-Int	Integer	-	0-1	-

From-Int

This indicates the sequence-number that is the lower bound for the range. If absent, the default is *no lower bound*; and the selection begins with the entry with the lowest sequence-number in the information base.

To-Int

This indicates the sequence-number that is the upper bound for the range. If absent, the default is *no upper bound*; and the selection finishes with the entry with the highest sequence-number in the information base.

8.2.54 Session

An instance of OM class **Session** provides information for a particular association between a client program and the Message Store.

An instance of this OM class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 8-39.

Table 8-39 Attributes Specific to Session

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
Initiator	Object(OR-Name)[1]	-	1	-
File-Descriptor[2]	Integer	-	0-1	(see below)

[1]As defined in Section 5.2.38, "OR Name," on page 5-58.

[2]The File-Descriptor attribute is not implemented in the current release of Solstice X.400

Initiator

This specifies the OR-Name of the initiator (the UA) of this session (or association) with the MS.

File-Descriptor

This indicates the file descriptor associated with the session. The file descriptor may be used in subsequent calls to vendor-specific system facilities to suspend the process (for example, UNIX System V `poll()` or Berkeley Source Distribution `select()`). Its use for other purposes is unspecified.

8.2.55 Strong-Credentials

An instance of OM class **Strong-Credentials** gives the credentials for strong authentication.

An instance of this OM class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 8-40.

Table 8-40 Attributes Specific to Strong-Credentials

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
Bind-Token	Object(Asymmetric-Token)[1]	-	0-1	-
Certificate	Object(Certificates)[2]	-	0-1	-

[1]As defined in Section 5.2.3, "Asymmetric Token," on page 5-5.

[2]As defined in the XDS Specification (see reference XDS in Appendix C, "Referenced Documents").

Bind-Token

This specifies the token used to convey to the protected security-relevant information.

Certificate

This specifies the certificate used to convey a verified copy of the public asymmetric encryption key of the subject of the certificate.

8.2.56 Submission Results

As defined in the Message Handling Package in Section 5.2.50, “Submission Results,” on page 5-70.

8.2.57 Submitted Communique

As defined in the Message Handling Package in Section 5.2.51, “Submitted Communique,” on page 5-71.

8.2.58 Submitted Message

As defined in the Message Handling Package in Section 5.2.52, “Submitted Message,” on page 5-74.

8.2.59 Submitted Probe

As defined in the Message Handling Package in Section 5.2.54, “Submitted Probe,” on page 5-78.

8.2.60 Summarize-Argument

An instance of OM class **Summarize-Argument** specifies the argument for the `ms_summarize` function.

An instance of this OM class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 8-41.

Table 8-41 Attributes Specific to Summarize-Argument

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
Information-Base-Type	Integer	-	0-1	stored-messages
Selector	Object(Selector)	-	1	-
Summary-Requests	Object(Summary-Requests)	-	0-1	-

Information-Base-Type

Specifies the repository containing entries for delivered messages and reports. Its value is **stored-messages** *{MS_STORED_MESSAGES}*.

Selector

This is the set of criteria for determining which entries shall be summarised.

Summary-Requests

This indicates the sequence of attribute-types for which summaries are requested.

8.2.61 Summary

An instance of OM class **Summary** describes part of the result returned by the `ms_summarize` function; it summarises the count of entries based on the presence (and then, based on the actual attribute-values) or absence of certain various attribute-types of the entries, as requested.

An instance of this OM class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 8-42.

Table 8-42 Attributes Specific to Summary

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
Absent	Integer	-	0-1	-
Present	Object(Summary-Present)	-	0/more	-

Absent

This is a count of the entries that do not contain an attribute of the attribute-type specified in the request. This is absent if there are no such entries.

Present

This gives a list of summaries for the entries that do contain an attribute of the attribute-type specified, broken down by the attribute-values actually present. This is absent if there are no such entries.

8.2.62 Summary-Present

An instance of OM class **Summary-Present** describes part of the result returned by the `ms_summarize` function; in particular, it gives the count of entries that contain attributes of the specified attribute-type for the specified attribute-value.

An instance of this OM class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 8-43.

Table 8-43 Attributes Specific to Summary-Present

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
AVA	Object(AVA)	-	1	-
Count	Integer	-	0-1	-

AVA

This specifies an attribute value assertion (in other words, it specifies the attribute-type and a particular attribute-value). The `ms_summarize` function returns, in **Count**, the total of the entries satisfying this attribute value assertion.

Count

This is a count of the entries which do contain the attribute of the specified attribute-type and whose value matches the specified attribute-value. This is absent if there are no such entries.

8.2.63 Summary-Requests

An instance of OM class **Summary-Requests** describes the sequence of attribute-types for which summaries are requested in the `ms_summarize` function.

An instance of this OM class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 8-44.

Table 8-44 Attributes Specific to Summary-Requests

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
Requested-Str	String(Object-Identifier)	-	0/more	-

Requested-Str

This is the sequence of attribute-types for which summaries are requested. This is only present if a summary is requested.

8.2.64 Summary-Result

An instance of OM class **Summary-Result** gives the result of a successful `ms_summarize` function.

An instance of this OM class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 8-45.

Table 8-45 Attributes Specific to Summary-Result

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
Next	Integer	-	0-1	-
Count	Integer	-	1	-
Span	Object(Sequence-Number-Range)	-	0-1	-
Summaries	Object(Summary)	-	0/more	-

Next

This is returned in the case where the number of entries selected would have been greater if it were not for the limit specified in the selector. This would then give the sequence-number for the next entry that would have been selected.

Count

This gives the number of entries that matched the selection criteria.

Span

This gives the range of sequence-numbers of entries that matched the selection criteria. It is absent if there were no such entries (in other words, **Count** value is zero).

Summaries

This is a sequential list of summaries; one for each summary-request. The summaries are returned in the order that they were requested in the `ms_summarize` function.

8.2.65 Wait-Result

An instance of OM class **Wait-Result** gives the result of a successful `ms_wait` function.

An instance of this OM class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 8-46.

Table 8-46 Attributes Specific to Wait-Result

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
Wait-New-Available	Boolean	-	1	true

Wait-New-Available

This indicates whether any new MS entries have become available for retrieval. Its value is true if new entries in the MS have become available for retrieval, and false if otherwise.

Message Store Package Errors 9

This chapter defines the errors that can arise in the use of the MS interface and describes the method used to report them.

Errors are reported to the application program by means of the **Status** that is returned by most functions (it is the function result in the C language binding). A function which completes successfully returns the value Success *{MS_SUCCESS}*. When a function is not successful it will return a private object that contains details about the problem which prevented its successful completion, unless no workspace exists (that is, if `ms_initialize` has not been called successfully) in which case the function returns the error constant **no-workspace** *[MS_NO_WORKSPACE]*.

There are two types of failure for asynchronous operations. The first type is reported immediately in the status of the invoking function whereas the second type is returned as the **Operation-Status** result of a later call to `ms_receive_result`. However, the distinction between these two types of failure is implementation-dependent.

Errors are classified into several OM classes. The Standards (see reference X.413 in Appendix C, “Referenced Documents”) classify errors into several different kinds.

The MS interface also defines three more kinds of error:

- Library-Error
- Communications-Error
- System-Error.

Each of these is represented by an OM class and are described in the next section. All of them inherit the OM attribute **Problem** from their superclass *Error*, which is described first.

All the OM classes defined in this chapter are part of the Message Store package introduced in Chapter 8, “Message Store Package”. The possible errors that each MS interface function may generate are listed for the respective interface functions in Chapter 4, “Message Store (MS) Interface”. Errors are mandatory only if the applicable FU is supported.

9.1 MS Errors Class Hierarchy

This section depicts the hierarchical organisation of the classes defined in this chapter and thus, indicates the inheritance of additional OM attributes from its superclass(es) by each class. Subclassification is indicated by indentation and the names of abstract classes are rendered in italics. Thus, for instance, the concrete class **Attribute-Problem** is an immediate subclass of the abstract class *Error* which in turn is an immediate subclass of the abstract class *Object*.

Object (see the *Solstice XOM Programming Reference Manual*)

- Attribute-Error
- Auto-Action-Request-Error
- Delete-Error
- Fetch-Restriction-Error
- Sequence-Number-Error
- *Error*
  — Attribute-Problem
  — Auto-Action-Request-Problem
  — Bind-Error
  — Cancel-Submission-Error
  — Communications-Error

— Delete-Problem
— Element-Of-Service-Not-Subscribed-Error
— Fetch-Restriction-Problem
— Inconsistent-Request-Error
— Invalid-Parameters-Error
— Library-Error
— Originator-Invalid-Error
— Range-Error
— Recipient-Improperly-Specified-Error
— Register-Rejected-Error
— Remote-Bind-Error
— Security-Error
— Service-Error
— Sequence-Number-Problem
— Submission-Control-Violated-Error
— System-Error
— Unsupported-Critical-Function-Error

The client program is not permitted to create or modify any instances of these OM classes. Furthermore, this specification does not mandate that any OM classes be translatable using `OM_Encode()` and `OM_Decode()`.

Note – An **Attribute-Error**, not a subclass of Error, is special in that it may report several problems at a time; each problem is reported in an **Attribute-Problem**, which is a subclass of Error. The situation is similar for **Auto-Action-Request-Error**, **Delete-Error** and **Fetch-Restriction-Error** along with their corresponding **Problem** classes.

9.2 MS Errors Class Definitions

This section defines the error classes of the MS package. Other classes are defined in Chapter 8, “Message Store Package”.

9.2.1 Error

The OM class **Error** consists of parameters common to all errors.

This OM class is an abstract class which has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 9-1.

Table 9-1 Attributes Specific to Error

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
Problem	Enum(Problem)	-	1	-

Problem

This indicates the type of error. A number of possible values are defined but implementations may define additional values. Implementations will not return other values for error conditions described in this chapter. Each of the standard values listed below is described under the relevant error OM class to which it applies. The possible values are:

action-type-not-subscribed	invalid-feature
asynchrony-not-supported	invalid-parameters
attribute-type-not-subscribed	message-submission-identifier-invalid
authentication-error	miscellaneous
bad-argument	no-such-class
bad-class	no-such-entry
bad-session	originator-invalid
busy	out-of-memory
child-entry-specified	register-rejected
communications-problem	remote-bind-error
content-length-problem	reversed

content-type-problem	security
deferred-delivery-cancellation-rejected	submission-control-violated
delete-restriction-problem	too-many-operations
eit-problem	too-many-sessions
element-of-service-not-subscribed	unable-to-establish-association
feature-not-negotiated	unacceptable-security-context
feature-unavailable	unavailable
inappropriate-for-operation	unavailable-action-type
inappropriate-matching	unavailable-attribute-type
inconsistent-request	unsupported-critical-function
invalid-attribute-value	unwilling-to-perform

9.2.2 Attribute-Error

An instance of OM class **Attribute-Error** reports one or more attribute-related problems encountered while performing a function as requested on a particular occasion (see Section 9.2 in X.413).

An instance of this OM class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 9-2.

Table 9-2 Attributes Specific to Attribute-Error

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
Attribute-Problems	Object(Attribute-Problem)	-	1/more	-

Attribute-Problems

This provides information about the attribute-related problem. An **Attribute-Error** can report several problems at a time.

9.2.3 Attribute-Problem

An instance of OM class **Attribute-Problem** documents one attribute-related problem encountered while performing a function as requested on a particular occasion.

An instance of this OM class has the OM attributes of its superclasses *Object*, *Error* and additionally, the OM attributes listed in Table 9-3.

Table 9-3 Attributes Specific to Attribute-Problem

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
Attribute-Type	String(Object-Identifier)	-	1	-
Attribute-Value	any	-	0-1	-

Attribute-Type

This indicates the type of the attribute with which the problem is associated.

Attribute-Value

This indicates the value of the attribute with which the problem is associated.

Additional Constraints

The OM attribute **Problem** is inherited from the superclass *Error*. Its value is one of:

- **attribute-type-not-subscribed**
  means that an attribute-type used as argument to the function is not one of those to which the client has subscribed.
- **inappropriate-for-operation**
  means that an attribute-type used as an argument of the function is unsuitable for its required use.
- **inappropriate-matching**
  means that the filter contains a filter-item in which an attribute is matched using an operation (equality, ordering or substrings) that is not defined for that attribute.
- **invalid-attribute-value**
  means that a purported attribute-value specified as an argument of the function does not conform to the data-type for the attribute-type concerned.
- **unavailable-attribute-type**
  means that a purported attribute-type used as an argument of the function is not one of those which is supported by the MS. If the MS is able to carry out the operation anyway, it is not prohibited from so doing.

9.2.4 *Auto-Action-Request-Error*

An instance of OM class **Auto-Action-Request-Error** reports one or more problems related to registration of an auto-action.

An instance of this OM class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 9-4.

Table 9-4 Attributes Specific to Auto-Action-Request-Error

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
Auto-Action-Request-Problems	Object(Auto-Action-Request-Problem)	-	1/more	-

Auto-Action-Request-Problems

This indicates type of problems that apply to the given auto-actions. An **Auto-Action-Request-Error** can report several problems at a time.

9.2.5 *Auto-Action-Request-Problem*

An instance of OM class **Auto-Action-Request-Problem** documents the type of problem encountered when attempting to register an auto-action.

An instance of this OM class has the OM attributes of its superclasses: *Object*, *Error* and additionally, the OM attributes listed in Table 9-5.

Table 9-5 Attributes Specific to Auto-Action-Request-Problem

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
Type	String(Object-Identifier)	-	1	-

Type

This identifies the auto-action type for which a problem is encountered when attempting to register it.

Additional Constraints

The OM attribute **Problem** is inherited from the superclass *Error*. Its value is one of:

- **action-type-not-subscribed**
  means that an action-type used as argument to the function is not one of those to which the client has subscribed.
- **unavailable-action-type**
  means that the action-type used as an argument of the function is not one of those supported by the MS.

9.2.6 Bind-Error

An instance of OM class **Bind-Error** indicates an error associated with the `ms_bind` function.

An instance of this OM class has the OM attributes of its superclass *Object*, *Error* and no additional OM attributes.

Additional Constraints

The OM attribute **Problem** is inherited from the superclass *Error*. Its value is one of:

- **authentication-error**
  means that the `ms_bind` function failed because the client's **Credentials** are not acceptable or are improperly specified.
- **unable-to-establish-association**
  means that the MS has rejected the client's attempt to establish an association with the `ms_bind` function.
- **unacceptable-security-context**
  means that the `ms_bind` function failed because the **Security-Context** proposed by the client for the `ms_bind` function is unacceptable to the MS.

9.2.7 Cancel-Submission-Error

An instance of OM class **Cancel-Submission-Error** indicates a disruption in the performance of the `ms_cancel_submission` function.

An instance of this OM class has the OM attributes of its superclasses *Object*, *Error* and no additional OM attributes.

Additional Constraints

The OM attribute **Problem** is inherited from the superclass *Error*. Its value is one of:

- **deferred-delivery-cancellation-rejected**
  means that the MTS cannot cancel the deferred delivery of a message, either because the message has already been processed for transfer and/or delivery, or because the MTS had provided the originator with proof-of-submission.
- **message-submission-identifier-invalid**
  means that the deferred delivery of a message cannot be cancelled due to the invalid message-submission-identifier specified.
- **remote-bind-error**
  means that the function requested cannot be performed because the MS is unable to bind to the MTS.

9.2.8 Communications-Error

An instance of OM class **Communications-Error** reports an error occurring in the other communications services supporting the MS. Such errors may include those arising in Remote Operation, Association Control, Presentation, Session and Transport.

An instance of this OM class has the OM attributes of its superclasses *Object*, *Error* and no additional OM attributes.

Additional Constraints

The OM attribute **Problem** is inherited from the superclass *Error*. Its value is:

- **communications-problem**
  means that an error has occurred in the other communications services supporting the MS. Such errors include those arising in Remote Operation, Association Control, Presentation, Session and Transport.

9.2.9 Delete-Error

An instance of OM class **Delete-Error** reports one or more problems encountered while attempting to delete one or more entries from an information base.

An instance of this OM class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 9-6.

Table 9-6 Attributes Specific to Delete-Error

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
Delete-Problems	Object(Delete-Problem)	-	1/more	-

Delete-Problems

This provides information about problems encountered when attempting to delete one or more entries in an information base. Any number of individual problems may be indicated, each problem is reported along with the sequence-number of the entry causing the problem. A **Delete-Error** can report several problems at a time.

9.2.10 Delete-Problem

An instance of OM class **Delete-Problem** documents the type of problem encountered when attempting to delete an entry in an information base and also identifies the sequence-number of the entry.

An instance of this OM class has the OM attributes of its superclasses *Object*, *Error* and additionally, the OM attributes listed in Table 9-7.

Table 9-7 Attributes Specific to Delete-Problem

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
Sequence-Number	Integer	-	1	-

Sequence-Number

This identifies the sequence-number of the entry causing the problem whose value is specified in the OM attribute **Problem**.

Additional Constraints

The OM attribute **Problem** is inherited from the superclass *Error*. Its value is one of:

- **child-entry-specified**
  means that an attempt has been made to delete a child-entry.

- **delete-restriction-problem**
  means that an attempt has been made to violate a restriction specified for the `ms_delete` function.

9.2.11 Element-Of-Service-Not-Subscribed-Error

An instance of OM class **Element-Of-Service-Not-Subscribed-Error** reports that the requested operation cannot be provided by the MTS because the MS has not subscribed to one of the elements of service the request requires.

An instance of this OM class has the OM attributes of its superclasses *Object*, *Error* and no additional OM attributes.

Additional Constraints

The OM attribute **Problem** is inherited from the superclass *Error*. Its value is:

- **element-of-service-not-subscribed**
  means that the requested operation cannot be provided by the MTS because the MS has not subscribed to one of the elements of service the request requires.

9.2.12 Fetch-Restriction-Error

An instance of OM class **Fetch-Restriction-Error** reports an attempt to violate a restriction associated with the `ms_fetch` function.

An instance of this OM class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 9-8.

Table 9-8 Attributes Specific to Fetch-Restriction-Error

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
Fetch-Restriction-Problems	Object(Fetch-Restriction-Problem)	-	1/more	-

Fetch-Restriction-Problems

This provides information about an attempt to violate a restriction associated with the `ms_fetch` function. A **Fetch-Restriction-Error** can report several problems at a time.

9.2.13 Fetch-Restriction-Problem

An instance of OM class **Fetch-Restriction-Problem** reports on a Fetch restriction violation problem that was encountered.

An instance of this OM class has the OM attributes of its superclasses *Object*, *Error* and additionally, the OM attributes listed in Table 9-9.

Table 9-9 Attributes Specific to Fetch-Restriction-Problem

OM Attribute	**Value Syntax**	**Value Length**	**Value Number**	**Value Initially**
Content-Type-Int	Integer	-	0-1	-
Content-Type	String(Object-Identifier)	-	0-1	-
EITs	String(Object-Identifier)	-	0/more	-
Content-Length	Integer	-	0-1	-

Content-Type-Int

Content-Type

These indicate the content-type of the message with which the Fetch restriction problem is associated.

EITs

This indicates the encoded-information-types requested in the `ms_fetch` function that are disallowed by the Fetch restrictions currently in effect.

Content-Length

This indicates the content-length of the message with which the Fetch restriction problem is associated.

Additional Constraints

The OM attribute **Problem** is inherited from the superclass *Error*. Its value is one of:

- **content-length-problem**
  means that the content-length of the message being fetched exceeds that permitted by the Fetch restrictions currently in effect.

- **content-type-problem**
  means that the content-type of the message being fetched is disallowed by the Fetch restrictions currently in effect.
- **eit-problem**
  means that the encoded-information-types requested in the `ms_fetch` function are disallowed by the Fetch restrictions currently in effect.

9.2.14 Inconsistent-Request-Error

An instance of OM class **Inconsistent-Request-Error** reports a problem where the requested operation cannot be provided by the MTS because the MS has made an inconsistent request.

An instance of this OM class has the OM attributes of its superclasses *Object*, *Error* and no additional OM attributes.

Additional Constraints

The OM attribute **Problem** is inherited from the superclass *Error*. Its value is:

- **inconsistent-request**
  means that the requested operation cannot be provided by the MTS because the MS has made an inconsistent request.

9.2.15 Invalid-Parameters-Error

An instance of OM class **Invalid-Parameters-Error** indicates a semantic problem in the set of parameters received for a function. This error would be used, for example, to report that an optional parameter was present in the wrong context, or to report that a value for one of the parameters is inappropriate.

An instance of this OM class has the OM attributes of its superclasses *Object*, *Error* and no additional OM attributes.

Additional Constraints

The OM attribute **Problem** is inherited from the superclass *Error*. Its value is:

- **invalid-parameters**
  means that there is a semantics problem in the set of parameters received for a function; this error would be used, for example, to report that an optional parameter was present in the wrong context, or to report that a value for one of the parameters is inappropriate.

9.2.16 Library-Error

An instance of OM class **Library-Error** reports an error detected by the interface function library.

An instance of this OM class has the OM attributes of its superclasses *Object*, *Error* and no additional OM attributes.

Each function has several possible errors which can be detected by the interface library itself and which are returned directly by the function call. These errors occur when the library itself is capable of performing an action, submitting a service request or deciphering a response from the MS.

Additional Constraints

The OM attribute **Problem** is inherited from the superclass *Error*. Its value is one of:

- **asynchrony-not-supported**
  means that the client requested the function be performed asynchronously when the service does not support asynchronous operations.
- **bad-argument**
  means that a bad argument was supplied; for example, using an instance of an OM class **Attribute** with no values of the OM attribute **Attribute-Value** as an input argument to a function will result in this error (since every MS attribute always has at least one value).
- **bad-class**
  means that the OM class of an argument is not supported for this function.

- **bad-session**
  means that there is no such session over which to present the function.
- **feature-unavailable**
  means that either the function to be invoked or the OM class of some argument supplied is not supported by the service.
- **feature-not-negotiated**
  means that either the functional unit(s) negotiated at `ms_initialize` does include this function being invoked or the package(s) negotiated at `ms_initialize` does include the OM class of some argument supplied.
- **miscellaneous**
  means that a miscellaneous error has occurred. This error will be returned if the interface cannot clear a transient system error by retrying the affected system call.
- **no-such-class**
  means that the argument supplied has a subobject which is an instance of a class not in the currently negotiated package(s).
- **out-of-memory**
  means that no more memory can be allocated.
- **too-many-operations**
  means that no more functions can be performed until at least one asynchronous operation has completed.
- **too-many-sessions**
  means that no more sessions can be bound with the message store until at least one existing session has unbound.

9.2.17 Originator-Invalid-Error

An instance of OM class **Originator-Invalid-Error** reports that the communique cannot be submitted because the originator was incorrectly identified.

An instance of this OM class has the OM attributes of its superclasses *Object*, *Error* and no additional OM attributes.

Additional Constraints

The OM attribute **Problem** is inherited from the superclass *Error*. Its value is:

- **originator-invalid**
  means that the communique cannot be submitted because the originator was incorrectly identified.

9.2.18 Range-Error

An instance of OM class **Range-Error** reports a problem related to the limit specified in a selector as an argument to a function.

An instance of this OM class has the OM attributes of its superclasses *Object*, *Error* and no additional OM attributes.

Additional Constraints

The OM attribute **Problem** is inherited from the superclass *Error*. Its value is:

- **reversed**
  means that the upper bound indicated a sequence-number or creation-time before that indicated by the lower bound.

9.2.19 Recipient-Improperly-Specified-Error

An instance of OM class **Recipient-Improperly-Specified-Error** gives the recipient names that were improperly specified in a communique presented for submission.

An instance of this OM class has the OM attributes of its superclasses *Object*, *Error* and additionally, the OM attributes listed in Table 9-10.

Table 9-10 Attributes Specific to Recipient-Improperly-Specified-Error

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
Recipients	Object(OR-Name)[1]	-	1/more	-

[1] As defined in Section 5.2.38, “OR Name,” on page 5-58.

Recipients

This lists the improperly specified recipients of the communique presented for submission.

Additional Constraints

The OM attribute **Problem** is inherited from the superclass *Error*. Its value is:

- **recipient-improperly-specified**
  means that the communique cannot be submitted because it is directed to improperly specified recipients.

9.2.20 Register-Rejected-Error

An instance of OM class **Register-Rejected-Error** indicates that the `ms_register` function has been rejected.

An instance of this OM class has the OM attributes of its superclasses *Object*, *Error* and no additional OM attributes.

Additional Constraints

The OM attribute **Problem** is inherited from the superclass *Error*. Its value is:

- **register-rejected**
  means that the `ms_register` function has been rejected.

9.2.21 Remote-Bind-Error

An instance of OM class **Remote-Bind-Error** reports that the requested function cannot be provided by the MS because the MS is unable to bind to the MTS.

An instance of this OM class has the OM attributes of its superclasses *Object*, *Error* and no additional OM attributes.

Additional Constraints

The OM attribute **Problem** is inherited from the superclass *Error*. Its value is:

- **remote-bind-error**
  means that requested function cannot be provided by the MS because the MS is unable to bind to the MTS.

9.2.22 Security-Error

An instance of OM class **Security-Error** reports that the requested operation could not be provided by the MTS because it would violate the security policy in effect.

An instance of this OM class has the OM attributes of its superclasses *Object*, *Error* and additionally, the OM attributes listed in Table 9-11.

Table 9-11 Attributes Specific to Security-Error

OM Attribute	**Value Syntax**	**Value Length**	**Value Number**	**Value Initially**
Security-Problem	Integer	-	1	-

Security-Problem

This identifies the cause of the violation of the security policy.

Additional Constraints

The OM attribute **Problem** is inherited from the superclass *Error*. Its value is:

- **security**
  means that the requested operation could not be provided by the MTS because it would violate the security policy in effect.

9.2.23 Sequence-Number-Error

An instance of OM class **Sequence-Number-Error** reports one or more problems related to the sequence-numbers specified as argument to a function.

An instance of this OM class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 9-12.

Table 9-12 Attributes Specific to Sequence-Number-Error

OM Attribute	**Value Syntax**	**Value Length**	**Value Number**	**Value Initially**
Sequence-Number-Problems	Object(Sequence-Number-Problem)	-	1/more	-

Sequence-Number-Problems

This gives the sequence-numbers specified as argument to a function that caused whatever problems. A **Sequence-Number-Error** can report several problems at a time.

9.2.24 Sequence-Number-Problem

An instance of OM class **Sequence-Number-Problem** documents the type of problem related to sequence-numbers specified as argument to a function.

An instance of this OM class has the OM attributes of its superclasses *Object*, *Error* and additionally, the OM attributes listed in Table 9-13.

Table 9-13 Attributes Specific to Sequence-Number-Problem

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
Sequence-Number	Integer	-	1	-

Sequence-Number

This identifies the sequence-number of the entry causing the problem.

Additional Constraints

The OM attribute **Problem** is inherited from the superclass *Error*. Its value is:

- **no-such-entry**
  means that the sequence-number supplied does not match that of any entry in the information base.

9.2.25 Service-Error

An instance of OM class **Service-Error** reports a problem related to MS service.

An instance of this OM class has the OM attributes of its superclasses *Object*, *Error* and no additional OM attributes.

Additional Constraints

The OM attribute **Problem** is inherited from the superclass *Error*. Its value is one of:

- **busy**
  means that the MS, or some part of it, is presently too busy to perform the requested function but may be able to do so after a short while.
- **unavailable**
  means that the MS, or some part of it, is presently unavailable.
- **unwilling-to-perform**
  means that the MS is not prepared to execute the requested function because it would lead to excessive consumption of resources.

9.2.26 Submission-Control-Violated-Error

An instance of OM class **Submission-Control-Violated-Error** reports the violation by the MS of a control on submission imposed by the MTS.

An instance of this OM class has the OM attributes of its superclasses *Object*, *Error* and no additional OM attributes.

Additional Constraints

The OM attribute **Problem** is inherited from the superclass *Error*. Its value is:

- **submission-control-violated**
  means that the MS has violated a control on submission imposed by the MTS.

9.2.27 System-Error

An instance of OM class **System-Error** reports an error occurring in the underlying operating system.

An instance of this OM class has the OM attributes of its superclasses *Object*, *Error* and no additional OM attributes.

Additional Constraints

The OM attribute **Problem** is inherited from the superclass *Error*. Its value is the same as that of *errno* defined in the C language.

The standard names of system errors are defined in the X/Open Portability Guide, Issue 4 (XPG4), System Interface and Headers (see reference XSH in Appendix C, “Referenced Documents”), and additional names may be defined by an implementation.

If a transient error occurs *[EINTR]* or *[EAGAIN]* implementations will retry the affected function and will not report these errors. If such an error persists, they may report the Library-Error (miscellaneous) or an implementation-defined library error.

9.2.28 Unsupported-Critical-Function-Error

An instance of OM class **Unsupported-Critical-Function-Error** reports that an argument of the function was marked as critical-for submission but is unsupported by the MTS.

An instance of this OM class has the OM attributes of its superclasses *Object*, *Error* and no additional OM attributes.

Additional Constraints

The OM attribute **Problem** is inherited from the superclass *Error*. Its value is:

- **unsupported-critical-function**
  means that an argument of the function was marked as **critical-for-submission** but is unsupported by the MTS.

Message Store General Attributes Package 10

This chapter defines the Message Store General Attributes Package. The MS standards define a number of MS attribute types, known as General-Attributes. They allow MS entries to be utilised. This chapter lists the names for each of these MS attribute types and defines OM classes to represent those which are not represented directly by OM syntaxes. The values of MS attributes are not restricted to those discussed in this chapter; new attribute types and syntaxes to be used in conjunction with an MS may be created in the future. Section 4.2.3.1, “Attribute,” on page 4-5 defines how the values of other syntaxes are represented in the interface.

The constants and OM classes defined in this chapter are additional to those in the MS Package defined in Chapter 8, “Message Store Package” and Chapter 9, “Message Store Package Errors”.

This Message Store General Attributes Package is mandatory in that the service is required to provide those classes which can be supported by the underlying implementation of the MS.

The object identifier associated with the Message Store General Attributes Package is represented by the constant:

Message-Store-General-Attributes-Package
{MS_GENERAL_ATTRIBUTES_PACKAGE}

See Table 1-3 on page 1-6 for the value of this object identifier.

10.1 MS GA Types

This section presents the MS attribute types defined in the Standards for use in MS entries. Each MS entry is composed of a number of MS attributes which comprises an attribute type along with one or more attribute values. The form of each value of an MS attribute is determined by the attribute syntax associated with the type of the attribute.

Note the distinction between MS attributes used with regard to the MS and OM attributes. The term *MS attribute* or the unqualified term *attribute* is used to denote the MS construct, whereas the phrase *OM attribute* is used to denote the Object Management one.

In the MS interface, MS attributes appear as instances of the OM class **Attribute** with the attribute type represented as the value of the OM attribute **Attribute-Type** and the attribute value(s) represented as the value(s) of the OM attribute **Attribute-Values**. Each attribute type has an object identifier, assigned by the Standards, which is the value of the OM attribute **Attribute-Type**. These object identifiers are represented in the interface by constants with the same name as the MS attribute prefixed with A- for ease of identification (and correspondingly, the C variables begin with *MS_A_*).

This section contains two tables.

- Table 10-1 tabulates the names of the MS attribute types defined in the Standards, together with the object identifiers associated with each of them.
- Table 10-2 on page 10-5 gives the names of the MS attribute types together with the OM Value Syntax used in the interface to represent values of that MS attribute type. This OM Value Syntax is the syntax of the OM attribute **Attribute-Values**. The table also includes:
  - An indication of whether the MS attribute can be multi-valued.

- Whether the attribute-type is always present (P), conditionally present (C), or absent in a delivered message entry, a delivered report entry, and a returned content entry.
- Whether or not the attribute-type can be returned in a list or an alert abstract-operation, and whether or not the attribute-type may be used in a summarize abstract-operation (for simple ASN.1 data-types).

Note that many of the OM Value Syntaxes are defined in Chapter 5, “Message Handling Packages”.

Table 10-1 Object Identifiers for MS Attribute Types

Attribute Type	**Object Identifier**[1]
Child-Sequence-Numbers	0
Content	1
Content-Confidentiality-Algorithm-Identifier	2
Content-Correlator	3
Content-Identifier	4
Content-Integrity-Check	5
Content-Length	6
Content-Returned	7
Content-Type	8
Conversion-With-Loss-Prohibited	9
Converted-EITs	10
Creation-Time	11
Delivered-EITs	12
Delivery-Flags	13
DL-Expansion-History	14
Entry-Status	15
Entry-Type	16
Intended-Recipient-Name	17
Message-Delivery-Envelope	18
Message-Delivery-Identifier	19

Table 10-1 Object Identifiers for MS Attribute Types *(Continued)*

Attribute Type	**Object Identifier**[1]
Message-Delivery-Time	20
Message-Origin-Authentication-Check	21
Message-Security-Label	22
Message-Submission-Time	23
Message-Token	24
Original-EITs	25
Originator-Certificate	26
Originator-Name	27
Other-Recipient-Names	28
Parent-Sequence-Number	29
Per-Recipient-Report-Delivery-Fields	30
Priority	31
Proof-Of-Delivery-Request	32
Redirection-History	33
Report-Delivery-Envelope	34
Reporting-DL-Name	35
Reporting-MTA-Certificate	36
Report-Origin-Authentication-Check	37
Security-Classification	38
Sequence-Number	39
Subject-Submission-Identifier	40
This-Recipient-Name	41

[1]These object identifiers are preceded by:
`{joint-iso-ccitt(2) mhs-motis(6) ms(4) attribute types(3)}`

Table 10-2 Value Syntax for MS Attribute Types

Attribute Type[1]	**OM Value Syntax**	**Presence in delivered message entry**	**Presence in delivered report entry**	**Presence in returned content entry**	**Avail. for list, alert**	**Avail. for summarize**
Child-Sequence-Numbers	Integer	C	C	C	Y	N
Content	Object(General-Content)	P	-	P	N	N
Content-Confidentiality-Algorithm-Id	String(Object-Identifier)	C	-	-	Y	N
Content-Correlator	any	-	C	-	Y	N
Content-Identifier	String(Printable)	C	C	-	Y	N
Content-Integrity-Check	Object(Algorithm-And-Result)	C	-	-	Y	N
Content-Length	Integer	P	-	P	Y	N
Content-Returned	Boolean	-	P	-	Y	Y
Content-Type	String(Object-Identifier)	P	C	C	Y	Y
Conversion-With-Loss-Prohibited	Boolean	C	-	-	Y	N
Converted-EITs	String(Object-Identifier)	C	-	-	Y	N
Creation-Time[2]	String(UTC-Time)	P	P	P	Y	N
Delivered-EITs	String(Object-Identifier)	P	-	-	Y	N
Delivery-Flags	Boolean	P	-	-	Y	N
DL-Expansion-History	Object(Expansion-Record)	C	C	-	Y	N
Entry-Status	Enum(Entry-Status)	P	P	P	Y	Y
Entry-Type	Enum(Entry-Type)	P	P	P	Y	Y
Intended-Recipient-Name	Object(OR-Name)	C	-	-	Y	N
Message-Delivery-Envelope	Object(Delivery-Envelope)	P	-	-	N	N
Message-Delivery-Identifier	Object(MTS-Identifier)	P	-	-	Y	N
Message-Delivery-Time[2]	String(UTC-Time)	P	-	-	Y	N
Message-Origin-Authentication-Check	Object(Algorithm-And-Result)	C	-	-	Y	N
Message-Security-Label	Object(Security-Label)	C	C	-	Y	N

Table 10-2 Value Syntax for MS Attribute Types *(Continued)*

Attribute Type[1]	**OM Value Syntax**	**Presence in delivered message entry**	**Presence in delivered report entry**	**Presence in returned content entry**	**Avail. for list, alert**	**Avail. for summarize**
Message-Submission-Time[2]	String(UTC-Time)	P	-	-	Y	N
Message-Token	Object(Asymmetric-Token)	C	-	-	Y	N
Original-EITs	String(Object-Identifier)	C	C	-	Y	N
Originator-Certificate	Object(Certificates)	C	-	-	Y	N
Originator-Name	Object(OR-Name)	P	-	-	Y	N
Other-Recipient-Names	Object(OR-Name)	C	-	-	Y	N
Parent-Sequence-Number	Integer	C	-	P	Y	N
Per-Recipient-Report-Delivery-Fields	Object(Delivered-Per-Recipient-Report)	-	P	-	Y	N
Priority	Enum(Priority)	P	-	-	Y	Y
Proof-Of-Delivery-Request	Boolean	C	-	-	Y	N
Redirection-History	Object(Redirection-Record)	C	-	-	Y	N
Report-Delivery-Envelope	Object(Deliver-Report)	-	P	-	N	N
Reporting-DL-Name	Object(OR-Name)	-	C	-	Y	N
Reporting-MTA-Certificate	Object(Certificates)	-	C	-	Y	N
Report-Origin-Authentication-Check	Object(Algorithm-And-Result)	C	C	-	Y	Y
Security-Classification	Enum(Security-Classification)	C	C	-	Y	Y
Sequence-Number	Integer	P	P	P	Y	N
Subject-Submission-Identifier	Object(MTS-Identifier)	-	P	-	Y	N
This-Recipient-Name	Object(OR-Name)	P	-	-	Y	N

[1]Attribute Types in italics are multi-valued.
[2]Attribute Type has a value length of 0-17.

10.2 MS GA Class Hierarchy

This section depicts the hierarchical organisation of the classes defined in this chapter and thus shows which classes inherit additional OM attributes from their superclasses. Subclassification is indicated by indentation and the names of abstract classes are rendered in italics. Thus, for instance, the concrete class **Delivered-Per-Recipient-NDR** is an immediate subclass of the abstract class *Delivered-Per-Recipient-Report* which in turn is an immediate subclass of the abstract class *Object*.

All MS GA classes (except *Object* and Certificates) are defined in Chapter 5, “Message Handling Packages”.

Object (see the *Solstice XOM Programming Reference Manual*)

- Algorithm
  - Algorithm and Result
- Asymmetric Token
- Certificates (defined in the XDS Specification)
- General Content
- Delivered Report
- Expansion Record
- MTS Identifier
- OR Name
- *Delivered Per-recipient Report*
  - Delivered Per-recipient Delivery Report
  - Delivered Per-recipient Non-Delivery Report
- Redirection Record
- Security Label

10.3 MS GA Syntax Definitions

This section defines the MS GA class enumeration syntaxes. These are the syntaxes in the Enumeration group specific to MS General Attributes.

10.3.1 Entry-Status

An instance of the enumeration syntax **Entry-Status** indicates the current processing status of an MS entry. Its value is chosen from the following:

- **new**
  The message has neither been accessed via the `ms_list` function nor has it been automatically processed by the MS.
- **listed**
  The message has been accessed via the `ms_list` function or the `ms_fetch` function but it has yet to be completely "processed."
- **processed**
  The message has been "completely fetched" or the MS has performed some automatic action on it. The exact definition of "completely fetched" is content-specific and is defined by the corresponding content-specific standards. The Entry-Status of a (non)-delivery notification becomes **processed** when the delivered report envelope is retrieved.

10.3.2 Entry-Type

An instance of the enumeration syntax **Entry-Type** indicates the type of an MS entry. Its value is chosen from the following:

- **delivered-message**
- **delivered-report**
- **returned-content**

10.3.3 Priority

An instance of the enumeration syntax **Priority** indicates the priority of the delivered message. Its value is chosen from the following:

- **normal**
- **low**
- **urgent**

10.3.4 Security-Classification

An instance of the enumeration syntax **Security-Classification** indicates security classification of a message. Its value is chosen from the following:

- **unmarked**
- **unclassified**
- **restricted**
- **confidential**
- **secret**
- **top-secret**

Message Store IM Attributes Package 11

The MS interface may be used with MS attributes other than those in the mandatory Message Store General Attributes Package described in Chapter 10, "Message Store General Attributes Package". This chapter describes one such optional package, the *Message Store Interpersonal Messaging (MS IM) Attributes Package*.

The MS IM Attributes Package contains all definitions for the Interpersonal Messaging MS attributes as defined in Annex J of reference X.420. Note that these attributes referred to as the Interpersonal Messaging MS Attributes in the X.420 will be known as the MS IM Attributes in this chapter.

This chapter lists each of these MS IM attribute types and defines OM classes to represent those which are not represented directly by OM syntaxes.

The *MS IM Attributes Package* is optional. The constants and OM classes defined in this chapter are additional to those in the MS Package (see Chapter 8, "Message Store Package" and Chapter 9, "Message Store Package Errors"), and they are not essential to the working of the interface but instead allow the access of the MS IM attributes.

The object identifier associated with the MS IM Attributes Package is represented by the constant:

Message-Store-Interpersonal-Messaging-Attributes-Package
{MS_IM_ATTRIBUTES_PACKAGE}.

See Table 1-4 on page 1-7 for the value of its object identifier.

The Object Management concepts and notation used in this chapter are introduced in Section 1.5, “Object Management”, of this document and are fully explained in the *Solstice XOM Programming Reference Manual*. A complete explanation of the meaning of the MS IM attributes and object classes is not given since this is outside the scope of this document (for details, see reference X.420). The purpose here is simply to present the representation of these items in the interface.

11.1 MS IM Attribute Types

This section presents the MS Interpersonal Messaging attribute types defined in the X.420 standards for use in MS entries. Each MS entry is composed of a number of MS attributes which comprise an attribute type and one or more attribute values. The form of each value of an MS IM attribute is determined by the attribute syntax associated with the type of the attribute.

Note the distinction between MS attributes used with regard to the MS and OM attributes. The term *MS attribute* or the unqualified term *attribute* is used to denote the MS construct, whereas the phrase *OM attribute* is used to denote the Object Management one.

In the interface, MS IM attributes appear as instances of the OM class **Attribute** with the attribute type represented as the value of the OM attribute **Attribute-Type** and the attribute value(s) represented as the value(s) of the OM attribute **Attribute-Values**. Each attribute type has an object identifier, assigned by the Standards, which is the value of the OM attribute **Attribute-Type**. These object identifiers are represented in the interface by constants with the same name as the MS IM attribute prefixed with IM- for ease of identification (and correspondingly, the C variables begin with *MS_IM_*).

This section contains two tables.

- Table 11-1 tabulates the names of the MS IM attribute types defined in Annex J of X.420, together with their respective object identifiers.

- Table 11-2 on page 11-5 gives the names of the MS IM attribute types together with the OM Value Syntax used in the interface. The table also includes:
  - An indication of whether the MS attribute can be multi-valued.
  - Whether the attribute-type is mandatory (M), conditionally present (C), or absent in a delivered message entry.
  - Whether or not the attribute-type is available for a list, alert, and summarize abstract-operation.
  - The range of lengths permitted for the string types.

The OM Value Syntax is the syntax of the OM attribute **Attribute-Values**. Note that many of the OM Value Syntaxes are defined in Chapter 6, "Interpersonal Messaging Packages".

Table 11-1 Object Identifiers for MS IM Attribute Types

MS IM Attributes	**Object Identifier**[1]
IM-Acknowledgment-Mode	9,9
IM-Authorizing-Users	7,10
IM-Auto-Forward-Comment	9,6
IM-Auto-Forwarded	7,9
IM-Bilaterally-Defined-Body-Parts	8,10
IM-Blind-Copy-Recipients	7,13
IM-Body	8,0
IM-Conversion-EITs	9,3
IM-Copy-Recipients	7,12
IM-Discard-Reason	9,5
IM-Expiry-Time	7,5
IM-Extended-Body-Part-Types	8,12
IM-G3-Fax-Body-Parts	8,3
IM-G3-Fax-Data	8,22
IM-G3-Fax-Parameters	8,15
IM-G4-Class1-Body-Parts	8,4
IM-Heading	7,0

Table 11-1 Object Identifiers for MS IM Attribute Types *(Continued)*

MS IM Attributes	**Object Identifier**[1]
IM-IA5-Text-Body-Parts	8,1
IM-IA5-Text-Data	8,20
IM-IA5-Text-Parameters	8,13
IM-Importance	7,7
IM-Incomplete-Copy	7,17
IM-IPM-Entry-Type	6,0
IM-IPM-Preferred-Recipient	9,2
IM-IPM-Synopsis	6,1
IM-IPN-Originator	9,1
IM-Languages	7,18
IM-Message-Body-Parts	8,8
IM-Message-Data	8,26
IM-Message-Parameters	8,19
IM-Mixed-Mode-Body-Parts	8,9
IM-Nationally-Defined-Body-Parts	8,11
IM-Non-Receipt-Reason	9,4
IM-NRN-Requestors	7,20
IM-Obsoleted-IPMs	7,14
IM-Originator	7,2
IM-Primary-Recipients	7,11
IM-Receipt-Time	9,8
IM-Related-IPMs	7,15
IM-Replied-To-IPM	7,3
IM-Reply-Recipients	7,16
IM-Reply-Requestors	7,21
IM-Reply-Time	7,6
IM-Returned-IPM	9,7

Table 11-1 Object Identifiers for MS IM Attribute Types *(Continued)*

MS IM Attributes	Object Identifier[1]
IM-RN-Requestors	7,19
IM-Sensitivity	7,8
IM-Subject	7,4
IM-Subject-IPM	9,0
IM-Suppl-Receipt-Info	9,10
IM-Teletex-Body-Parts	8,5
IM-Teletex-Data	8,23
IM-Teletex-Parameters	8,16
IM-This-IPM	7,1
IM-Videotex-Body-Parts	8,6
IM-Videotex-Data	8,24
IM-Videotex-Parameters	8,17

[1]These object identifiers are preceded by:
`{joint-iso-ccitt(2) mhs-motis(6) ipms(1)}`

Table 11-2 Value Syntax for MS IM Attribute Types

MS IM Attributes[1]	OM Value Syntax	Pres. in delivered message entry IPM	Pres. in delivered message entry NRN	Pres. in delivered message entry RN	Avail. for list, alert	Avail. for summarize
IM-Acknowledgment-Mode	Enum(Acknowledgment-Mode)	-	-	M	Y	Y
IM-Authorizing-Users	Object(OR-Descriptor)	C	-	-	Y	N
IM-Auto-Forward-Comment[3]	String(Printable)	-	C	-	Y	N
IM-Auto-Forwarded	Boolean	C	-	-	Y	Y
IM-Bilaterally-Defined-Body-Parts	Object(Bil-Defined-Body-Part)	C	-	-	N	N
IM-Blind-Copy-Recipients	Object(Recipient-Specifier)	C	-	-	Y	N
IM-Body	Object(Body)	M	-	-	N	N

Table 11-2 Value Syntax for MS IM Attribute Types *(Continued)*

MS IM Attributes[1]	**OM Value Syntax**	**Pres. in delivered message entry IPM**	**Pres. in delivered message entry NRN**	**Pres. in delivered message entry RN**	**Avail. for list, alert**	**Avail. for summarize**
IM-Conversion-EITs	String(Object-Identifier)	-	C	C	Y	N
IM-Copy-Recipients	Object(Recipient-Specifier)	C	-	-	Y	N
IM-Discard-Reason	Enum(Discard-Reason)	-	C	-	Y	Y
IM-Expiry-Time[2]	String(UTC-Time)	C	-	-	Y	N
IM-Extended-Body-Part-Types	String(Object Identifier)	C	-	-	Y	Y
IM-G3-Fax-Body-Parts	Object(G3-Fax-Body-Part)	C	-	-	N	N
IM-G3-Fax-Data	Object(G3-Fax-Data)	C	-	-	N	N
IM-G3-Fax-Parameters	Object(G3-Fax-NBPs)	C	-	-	N	N
IM-G4-Class1-Body-Parts	Object(G4-Class1-Body-Part)	C	-	-	N	N
IM-Heading	Object(Heading)	M	-	-	N	N
IM-IA5-Text-Body-Parts	Object(IA5-Text-Body-Part)	C	-	-	N	N
IM-IA5-Text-Data	String(IA5)	C	-	-	N	N
IM-IA5-Text-Parameters	Enum(IA5-Repertoire)	C	-	-	N	N
IM-Importance	Enum(Importance)	C	-	-	Y	Y
IM-Incomplete-Copy	Boolean	C	-	-	Y	N
IM-IPM-Entry-Type	Enum(IPM-Entry-Type)	M	M	M	Y	Y
IM-IPM-Preferred-Recipient	Object(OR-Descriptor)	-	C	C	Y	N
IM-IPM-Synopsis	Object(IPM-Synopsis)	M	-	-	N	N
IM-IPM-Originator	Object(OR-Descriptor)	-	C	C	Y	N
IM-Languages[4]	String(Printable)	C	-	-	Y	N
IM-Message-Body-Parts	Object(Message-Body-Part)	C	-	-	N	N
IM-Message-Data	Object(Interpersonal-Message)	C	-	-	N	N
IM-Message-Parameters	Object(Delivery-Envelope)	C	-	-	N	N
IM-Mixed-Mode-Body-Parts	Object(Mixed-Mode-Body-Part)	C	-	-	N	N

Table 11-2 Value Syntax for MS IM Attribute Types *(Continued)*

MS IM Attributes[1]	OM Value Syntax	Pres. in delivered message entry IPM	Pres. in delivered message entry NRN	Pres. in delivered message entry RN	Avail. for list, alert	Avail. for summarize
IM-Nationally-Defined-Body-Parts	Object(Nat-Defined-Body-Part)	C	-	-	N	N
IM-Non-Receipt-Reason	Enum(Non-Receipt-Reason)	-	M	-	Y	Y
IM-NRN-Requestors	Object(OR-Descriptor)	C	-	-	Y	N
IM-Obsoleted-IPMs	Object(IPM-Identifier)	C	-	-	Y	N
IM-Originator	Object(OR-Descriptor)	C	-	-	Y	N
IM-Primary-Recipients	Object(Recipient-Specifier)	C	-	-	Y	N
IM-Receipt-Time[2]	String(UTC-Time)	-	-	M	Y	N
IM-Related-IPMs	Object(IPM-Identifier)	C	-	-	Y	N
IM-Replied-To-IPM	Object(IPM-Identifier)	C	-	-	Y	N
IM-Reply-Recipients	Object(OR-Descriptor)	C	-	-	Y	N
IM-Reply-Requestors	Object(OR-Descriptor)	C	-	-	Y	N
IM-Reply-Time[2]	String(UTC-Time)	C	-	-	Y	N
IM-Returned-IPM	Object(IPM-Identifier)	-	C	-	Y	N
IM-RN-Requestors	Object(OR-Descriptor)	C	-	-	Y	N
IM-Sensitivity	Enum(Sensitivity)	C	-	-	Y	Y
IM-Subject[5]	String(Teletex)	C	-	-	Y	N
IM-Subject-IPM	Object(IPM-Identifier)	-	M	M	Y	N
IM-Suppl-Receipt-Info[6]	String(Printable)	-	-	C	Y	N
IM-Teletex-Body-Parts	Object(Teletex-Body-Part)	C	-	-	N	N
IM-Teletex-Data	Object(Teletex-Data)	C	-	-	N	N
IM-Teletex-Parameters	Object(Teletex-Parameters)	C	-	-	N	N
IM-This-IPM	Object(IPM-Identifier)	M	-	-	Y	N
IM-Videotex-Body-Parts	Object(Videotex-Body-Part)	C	-	-	N	N

Table 11-2 Value Syntax for MS IM Attribute Types *(Continued)*

MS IM Attributes[1]	OM Value Syntax	Pres. in delivered message entry IPM	Pres. in delivered message entry NRN	Pres. in delivered message entry RN	Avail. for list, alert	Avail. for summarize
IM-Videotex-Data	String(Videotex)	C	-	-	N	N
IM-Videotex-Parameters	Enum(Videotex-Syntax)	C	-	-	N	N

[1]Attribute Types in italics are multi-valued.
[2]Attribute Type has a value length of 0-17.
[3]Attribute Type has a value length of 0-256.
[4]Attribute Type has a value length of 2-5.
[5]Attribute Type has a value length of 0-128.
[6]Attribute Type has a value length of 1-256.

11.2 MS IM Attributes Class Hierarchy

This section depicts the hierarchical organisation of the classes defined in this chapter and thus shows which classes inherit additional OM attributes from their superclasses. Subclassification is indicated by indentation and the names of abstract classes are in italics. Thus, for instance, the concrete class **Bilaterally-Defined-Body-Part** is an immediate subclass of the abstract class *Body-Part* which in turn is an immediate subclass of the abstract class *Object*.

MS IM classes marked with an asterisk (*) are defined in Chapter 6, “Interpersonal Messaging Packages”. MS IM classes marked with two asterisks (**) are defined in Chapter 5, “Message Handling Packages”.

Object (defined in the *Solstice XOM Programming Reference Manual*)

- *Body Part* (*)
  - — Bilaterally Defined Body Part (*)
  - — Body
  - — G3 Fax Body Part (*)
  - — G4 Class 1 Body Part (*)
  - — IA5 Text Body Part (*)
  - — Message Body Part (*)
  - — Mixed-mode Body Part (*)
  - — Nationally-Defined Body Part (*)
  - — Teletex Body Part (*)
    - • Teletex-Parameters
  - — Teletex-Data
  - — Videotex Body Part (*)
  - — Body-Part-Synopsis
  - — *Content* (**)
    - • Interpersonal Message (*)
    - • *Interpersonal Notification* (*)
      - — Non-receipt Notification (*)
      - — Receipt Notification (*)
  - — Extensible Object (**)
    - • Delivery Envelope (**)
  - — G3-Fax-Data
  - — G3 Fax NBPs (**)

- — Heading
- — IPM Identifier (*)
- — IPM-Synopsis
- — Message-Body-Part-Synopsis
- — Non-Message-Body-Part-Synopsis
- — OR Descriptor (*)
- — Recipient Specifier (*)

11.3 MS IM Attributes Class Definitions

This section defines the MS IM attributes classes.

11.3.1 Body

An instance of OM class **Body** gives all the body parts of a message.

An instance of this OM class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 11-3.

Table 11-3 Attributes Specific to Body

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
Body-Part	Object(Body-Part)	-	0/more	-

Body-Part

A body part of the message.

11.3.2 Body-Part-Synopsis

An instance of OM class **Body-Part-Synopsis** gives the synopsis for an individual body part.

An instance of this OM class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 11-4.

Table 11-4 Attributes Specific to Body-Part-Synopsis

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
Message-Body-Part-Synopsis	Object(Message-Body-Part-Synopsis)	-	0-1	-
Non-Message-Body-Part-	Object(Non-Message-Body-Part-	-	0-1	-
Synopsis	Synopsis)			

Note – No instance of this class will contain more than one of the above OM attributes.

Message-Body-Part-Synopsis

This is the synopsis of a body part that is of type Message.

Non-Message-Body-Part-Synopsis

This is gives the synopsis of a body part that is of type other than Message.

11.3.3 G3-Fax-Data

An instance of OM class **G3-Fax-Data** gives the image data of a G3 facsimile body part of an interpersonal message.

An instance of this OM class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 11-5.

Table 11-5 Attributes Specific to G3-Fax-Data

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
Images	String(Bit)	-	0 or more	-

Images

The G3 facsimile images.

11.3.4 Heading

An instance of OM class **Heading** gives all the heading fields of an interpersonal message.

An instance of this OM class has the OM attributes of its superclass *Object*, and additionally, the OM attributes listed in Table 11-6.

Table 11-6 Attributes Specific to Heading

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
This-IPM	Object(IPM-Identifier)	-	1	-
Originator	Object(OR-Descriptor)	-	0-1	-
Authorizing-Users	Object(OR-Descriptor)	-	0/more	-
Primary-Recipients	Object(Recipient-Specifier)	-	0/more	-
Copy-Recipients	Object(Recipient-Specifier)	-	0/more	-
Blind-Copy-Recipients	Object(Recipient-Specifier)	-	0/more	-
Replied-To-IPM	Object(IPM-Identifier)	-	0-1	-
Obsoleted-IPMs	Object(IPM-Identifier)	-	0/more	-
Related-IPMs	Object(IPM-Identifier)	-	0/more	-
Subject	String(Teletex)	0-128	0-1	-

Table 11-6 Attributes Specific to Heading *(Continued)*

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
Expiry-Time	String(UTC-Time)	0-17	0-1	-
Reply-Time	String(UTC-Time)	0-17	0-1	-
Reply-Recipients	Object(OR-Descriptor)	-	0/more	-
Importance	Enum(Importance)	-	1	normal
Sensitivity	Enum(Sensitivity)	-	0-1	-
Auto-Forwarded	Boolean	-	1	false
Extensions	Object(Attribute)	-	0/more	-

This-IPM

This identifies the interpersonal message.

Originator

This identifies the originator of the interpersonal message.

Authorizing-Users

This identifies zero or more users who authorised the origination of the interpersonal message.

Primary-Recipients

This identifies zero or more users and distribution lists who are the “primary recipients” of the interpersonal message.

Copy-Recipients

This identifies zero or more users and distribution lists who are the “copy recipients” of the interpersonal message.

Blind-Copy-Recipients

This identifies zero or more users and distribution lists who are the “blind copy recipients” of the interpersonal message. A blind copy recipient is not disclosed to primary and copy recipients.

Replied-To-IPM

This identifies the interpersonal message to which the present interpersonal message is a reply.

Obsoleted-IPMs

This identifies the interpersonal messages that the authorising users consider the present interpersonal message to obsolete.

Related-IPMs

This identifies the interpersonal messages that the authorising users consider related to the present interpersonal message.

Subject

This identifies the subject of the interpersonal message.

Expiry-Time

This identifies when the authorising users consider the interpersonal message to lose its validity.

Reply-Time

This identifies when the authorising users request (but do not demand) that any replies to the present interpersonal message be originated.

Reply-Recipients

This identifies zero or more users or distribution lists whom the authorising users request (but do not demand) be among the preferred recipients of any replies to the present interpersonal message.

Importance

This identifies the importance (low, normal or high) that the authorising users attach to the interpersonal message.

Sensitivity

This identifies the sensitivity (personal, private or company-confidential) that the authorising users attribute to the interpersonal message.

Auto-Forwarded

This indicates whether the interpersonal message is a result of auto-forwarding.

Extensions

This conveys information accommodated by no other heading field. Some extensions (e.g., Languages, Incomplete-Copy) are defined in X.420, Annex A.

11.3.5 IPM-Synopsis

An instance of OM class **IPM-Synopsis** describes the structure, characteristics, size and processing status of an interpersonal message at the granularity of individual body parts.

An instance of this OM class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed.

Table 11-7 Attributes Specific to IPM-Synopsis

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
Body-Part-Synopsis	Object(Body-Part-Synopsis)	-	0/more	-

Body-Part-Synopsis

This is synopsis for an individual body part.

11.3.6 Message-Body-Part-Synopsis

An instance of OM class **Message-Body-Part-Synopsis** gives the synopsis of a body part that is of type Message.

An instance of this OM class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 11-8.

Table 11-8 Attributes Specific to Message-Body-Part-Synopsis

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
Sequence-Number	Integer	-	1	-
Synopsis	Object(IPM-Synopsis)	-	1	-

Sequence-Number

This is the sequence number assigned by the MS to the entry that the Message body part represents.

Synopsis

This is the synopsis of the interpersonal message that forms the content of the message that contains the body part.

11.3.7 Non-Message-Body-Part-Synopsis

An instance of OM class **Non-Message-Body-Part-Synopsis** gives the synopsis of a body part that is of type other than Message.

An instance of this OM class has the OM attributes of its superclass *Object* and additionally, the OM attributes listed in Table 11-9.

Table 11-9 Attributes Specific to Non-Message-Body-Part-Synopsis

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
Type	String(Object-Identifier)	-	1	-
Parameters	Object(External)[1]	-	1	-
Size	Integer	-	1	-
Processed	Boolean	-	1	false

[1]As defined in the *Solstice XOM Programming Reference Manual*.

Type

This is the extended type of the body part (i.e., the Direct-reference component of its Data component).

Parameters

This is the format and control parameters of the body part (i.e., its Parameters component).

Size

This is size in octets of the encoding of the Encoding component of the body part's Data component when the BER of X.209 (see reference X.209 in Appendix C, "Referenced Documents") are followed.

Processed

This indicates whether or not the body part has been conveyed to the UA by means of the `ms_list` or the `ms_fetch` function.

11.3.8 Teletex-Data

An instance of OM class **Teletex-Data** gives the data of a Teletex body part of an interpersonal message.

An instance of this OM class has the OM attributes of its superclass *Object* and additionally the OM attributes listed in Table 11-10 below.

Table 11-10 Attributes Specific to Teletex-Data

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
Teletex-Document	String(Teletex)	-	0/more	-

Teletex-Document

A page of the Teletex document.

11.3.9 Teletex-Parameters

An instance of OM class **Teletex-Parameters** describes the parameters of a Teletex body part of an interpersonal message.

An instance of this OM class has the OM attributes of its superclasses *Object*, **Teletex-Body-Part** (defined in Chapter 6, "Interpersonal Messaging Packages") and additionally, the OM attributes listed in Table 11-11.

Table 11-11 Attributes Specific to Teletex-Parameters

OM Attribute	Value Syntax	Value Length	Value Number Value	Initially
Number-Of-Pages	Integer	-	0-1	-

Number-Of-Pages

This indicates the number of pages contained in a Teletex body part of an interpersonal message.

Additional Constraints

The inherited attribute **Teletex-Document** shall not be present.

11.4 MS IM Attributes Syntax Definitions

This section defines the MS IM class enumeration syntaxes, i.e., the syntaxes in the Enumeration group specific to Interpersonal Messaging MS attributes.

11.4.1 Acknowledgment-Mode

As defined in the Interpersonal Messaging Package in Section 6.3.1, “Acknowledgment Mode,” on page 6-24.

11.4.2 Discard-Reason

As defined in the Interpersonal Messaging Package in Section 6.3.2, “Discard Reason,” on page 6-24.

11.4.3 IA5-Repertoire

As defined in the Interpersonal Messaging Package in Section 6.3.3, “IA5 Repertoire,” on page 6-25.

11.4.4 Importance

As defined in the Interpersonal Messaging Package in Section 6.3.4, “Importance,” on page 6-25.

11.4.5 IPM-Entry-Type

An instance of the enumeration syntax **IPM-Entry-Type** identifies the type of Interpersonal Messaging entry. Its value is chosen from the following:

Table 11-12 IPM Entry Type Reason Values

IPM	The entry is an interpersonal message.
RN	The entry is a receipt notification.
NRN	The entry is a non-receipt notification.

11.4.6 Non-Receipt-Reason

As defined in the Interpersonal Messaging Package in Section 6.3.6, “Non-receipt Reason,” on page 6-26.

11.4.7 Sensitivity

As defined in the Interpersonal Messaging Package in Section 6.3.8, “Sensitivity,” on page 6-26.

11.4.8 Videotex-Syntax

As defined in the Interpersonal Messaging Package in Section 6.3.9, “Videotex Syntax,” on page 6-27.

Compiling and Linking Applications 12

This chapter summarizes the implementation-specific information you need to compile and link applications based on the interfaces and packages specified in this document.

12.1 Header Files to Include for Solstice X.400 Products

The Solstice X.400 Messaging Server is supplied with the MA, MT, and MS libraries.

The Solstice X.400 Client Toolkit is supplied with the MA and MS libraries.

Table 12-1 on page 12-2 shows the header files that applications using the MA, MT, and MS libraries should include. The header files must be included in the order shown.

Table 12-1 Header Files to Include for Solstice X.400 Products

Header File	Defines	Include for MA/MT	Include for MS
/opt/SUNWconn/include/mhs/xom.h	Object Management interface	Y	Y
/opt/SUNWconn/include/mhs/xmh.h	MA interface	Y	N
/opt/SUNWconn/include/mhs/xms.h	MS interface	N	Y
/opt/SUNWconn/include/mhs/xds.h	DS packages	N	Y
/opt/SUNWconn/include/mhs/xmhp.h	MH packages	Y	Optional[1]
/opt/SUNWconn/include/mhs/ximp.h	IM packages	Optional	N
/opt/SUNWconn/include/mhs/xedip.h	EDI messaging package	Optional	N
/opt/SUNWconn/include/mhs/xmsga.h	MS General Attributes package	N	Y
/opt/SUNWconn/include/mhs/xmsima.h	MS IM Attributes packages	N	Optional
/opt/SUNWconn/include/mhs/xmsedia.h	MS EDI messaging package	N	Optional

[1]If an application uses functions defined in the MS Submission FU and the MS Administration FU, it must also include the `xmhp.h` header file.

See Appendix A, "Header Files" for a list of the symbols which are defined in the header files.

12.2 Compiling and Linking Procedure

To compile and link an application developed using the MA interface and related packages, specify the following entries in your Makefile:

```
CFLAGS  +=  -I  /opt/SUNWconn/include/mhs
LDFLAGS +=  -L  /opt/SUNWconn/lib -lxma
```

To compile and link an application developed using the MT interface and related packages, specify the following entries in your Makefile:

```
CFLAGS  += -I /opt/SUNWconn/include/mhs
LDFLAGS += -L /opt/SUNWconn/lib -lxapia
```

To compile and link an application developed using the MS interface and related packages, specify the following entries in your Makefile:

```
CFLAGS  +=  -I  /opt/SUNWconn/include/mhs
LDFLAGS +=  -L  /opt/SUNWconn/lib -lxms
```

To run your MA, MT, and MS applications, add the following path to the environment variable in your shell initialization file:

```
LD_LIBRARY_PATH   /opt/SUNWconn/lib
```

Header Files A

A.1 Introduction

This appendix sets out the symbols which are defined in the C headers. Where the values of the symbols are indicated, the values are an integral part of the interface. Where a value is not given, the value on a particular system will be determined by the application or by an administrator.

Note – The header files included with the Solstice X.400 Messaging Server and the Solstice X.400 Client Toolkit provide the same functionality as those defined in the X/Open specifications.

A.2 xmh.h

The declarations contained in the header file xmh.h deal with the MA and MT interfaces. The symbols specified by these declarations are the only MA and MT symbols that the service makes visible to the client. All applications that include this header must first include the OSI-Abstract-Data Manipulation header xom.h.

```
#ident “@(#)xmh.h      1.5 22 Sep 1995 SMI”
/*
 * Copyright 22 Sep 1995 Sun Microsystems, Inc. All Rights Reserved
 * (c) Copyright 1993 Marben Produit All rights reserved
 *
 */

#ifndef _XMH_H
#define _XMH_H

#ifdef  __cplusplus
extern “C” {
#endif

/* 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;
```

```
/* Invoke id for asynchrone operations */

typedef OM_sint MH_invoke_id;

/* MA FUNCTIONS */

/* The format of the function declarations allows for a prototyping
   that complies either to C ANSI requirements or to non-C ANSI requirements,
   according to macros defined in the “ugnupdef” include file.
*/

/* Cancel a submission with deffered delivery */
OM_return_code
ma_cancel_submission (
       OM_private_object session ,
       OM_object         mts_identifier
);

/* Close a Message Access Session */
OM_return_code
ma_close (
       OM_private_object session
);

/* Conclude the delivery in progress */
OM_return_code
ma_finish_delivery (
        OM_private_object session ,
        OM_object         delivery_confirmations ,
        OM_object         non_delivery_reports
);

/* Conclude the retrieval in progress */
OM_return_code
ma_finish_retrieval (
        OM_private_object session ,
        OM_boolean        remove
);

/* Open a Message Access Session */
OM_return_code
ma_open (
        OM_object         user_address ,
        OM_string         client_name ,
        MH_feature        feature_list[] ,
        OM_private_object *session ,
        OM_workspace      *workspace
);

/* Count messages and reports in incoming queue */
OM_return_code
ma_size (
        OM_private_object session ,
        MH_object_count   *number
);

/* Access to message or report */
OM_return_code
ma_start_delivery (
        OM_private_object session ,
        OM_private_object *object
);

/* Selective access to message or report */
OM_return_code
ma_start_retrieval (
        OM_private_object  session ,
        MH_sequence_number minimum ,
        MH_sequence_number *selected ,
        OM_private_object  *object
);

/* Submit a communique to the MTS */
OM_return_code
ma_submit (
        OM_private_object session ,
        OM_object         communique ,
        OM_private_object *submission_results
);

/* Wait an incoming message or report */
OM_return_code
ma_wait (
        OM_private_object  session ,
        MH_interval        interval ,
        MH_sequence_number minimum_sequence_number ,
        OM_boolean         *available
);
```

```
/* MT FUNCTIONS */

/* Close a Message Transfer Session */
OM_return_code
mt_close (
         OM_private_object session
);

/* Conclude the transfer in in progress */
OM_return_code
mt_finish_transfer_in (
         OM_private_object session ,
         OM_private_object object ,
         OM_boolean        rmv
);

/* Open a Message Transfer Session */
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
);

/* Count messages and reports in incoming queue */
OM_return_code
mt_size (
        OM_private_object session ,
        MH_object_count   *unreserved
);

/* Access to message or report */
OM_return_code
mt_start_transfer_in (
        OM_private_object session ,
        OM_private_object *object ,
        OM_string         *identifier
);

/* Add a Communique to the Output Queue */
OM_return_code
mt_transfer_out (
        OM_private_object session ,
        OM_object         object
);

/* Wait an incoming message or report */
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        "\126\006\001\002\005\001"
#define OMP_O_MH_FE_SUBMISSION          "\126\006\001\002\005\002"
#define OMP_O_MH_FE_DELIVERY            "\126\006\001\002\005\003"
#define OMP_O_MH_FE_RETRIEVAL           "\126\006\001\002\005\004"
#define OMP_O_MH_FE_BASIC_TRANSFER      "\126\006\001\002\005\005"
#define OMP_O_MH_FE_TRANSFER_IN         "\126\006\001\002\005\006"
#define OMP_O_MH_FE_TRANSFER_OUT        "\126\006\001\002\005\007"
#define OMP_O_MH_FE_IM_84               "\126\006\001\002\005\010"
#define OMP_O_MH_FE_IM_88               "\126\006\001\002\005\011"
#define OMP_O_MH_FE_MH_84               "\126\006\001\002\005\012"
#define OMP_O_MH_FE_MH_88               "\126\006\001\002\005\013"
#define OMP_O_MH_FE_SM_88               "\126\006\001\002\005\014"
#define OMP_O_MH_FE_MULTIPLE_DELIVERY   "\126\006\001\002\005\015"
#define OMP_O_MH_FE_EDI                 "\126\006\001\002\007\001"

/* 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_INSTANCE   ((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 */
/* BEGIN MA ASYNCHRONE INTERFACE */
#define MH_COMPLETED_OPERATION        ((OM_uint) 1)
#define MH_OUTSTANDING_OPERATION      ((OM_uint) 2)
#define MH_NO_OUTSTANDING_OPERATION   ((OM_uint) 3)
/* END MA ASYNCHRONE INTERFACE */

#ifdef  __cplusplus
}
#endif

#endif  /* _XMH_H */
```

A.3 xmhp.h

The declarations contained in the header file xmhp.h define the portion of the C interface that deals with MH packages. The symbols specified by these declarations are the only MH symbols that the service makes visible to the client. All applications that include this header must first include the OSI-Abstract-Data Manipulation header xom.h and the xmh.h header.

Note that 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 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 name are excepted and are abbreviated as shown in Table A-1.

Table A-1 Abbreviations for xmhp.h Class Names

Name	**Abbreviation**
BILATERAL_INFORMATION	BILATERAL_INFO
DELIVERED	DELIV
CONFIRMATION	CONFIRM
CONFIRMATIONS	CONFIRMS
PER_RECIPIENT_	PER_RECIP_
DELIV_PER_RECIP_REPORT	DELIV_PER_RECIP_REP

```
#ident “@(#)xmhp.h    1.4 27 Sep 1995 SMI”
/*
 *
 * Copyright 27 Sep 1995 Sun Microsystems, Inc. All Rights Reserved
 * (c) Copyright 1993 Marben Produit All rights reserved
 *
 */

#ifndef _XMHP_H
#define _XMHP_H

#ifdef  __cplusplus
extern “C” {
#endif
```

```
/* BEGIN MH PORTION OF INTERFACE */

/* SYMBOLIC CONSTANTS */

/* Class */

#define OMP_O_MH_C_ALGORITHM            “\126\006\001\002\005\013\000”
#define OMP_O_MH_C_ALGORITHM_AND_RESULT “\126\006\001\002\005\013\001”
#define OMP_O_MH_C_ASYMMETRIC_TOKEN     “\126\006\001\002\005\013\002”
#define OMP_O_MH_C_BILATERAL_INFO       “\126\006\001\002\005\013\003”
#define OMP_O_MH_C_COMMUNIQUE           “\126\006\001\002\005\013\004”
#define OMP_O_MH_C_CONTENT              “\126\006\001\002\005\013\005”
#define OMP_O_MH_C_DELIV_MESSAGE        “\126\006\001\002\005\013\006”
#define OMP_O_MH_C_DELIV_PER_RECIP_DR   “\126\006\001\002\005\013\007”
#define OMP_O_MH_C_DELIV_PER_RECIP_NDR  “\126\006\001\002\005\013\010”
#define OMP_O_MH_C_DELIV_PER_RECIP_REP  “\126\006\001\002\005\013\011”
#define OMP_O_MH_C_DELIV_REPORT         “\126\006\001\002\005\013\012”
#define OMP_O_MH_C_DELIVERY_CONFIRM     “\126\006\001\002\005\013\013”
#define OMP_O_MH_C_DELIVERY_ENVELOPE    “\126\006\001\002\005\013\014”
#define OMP_O_MH_C_EITS                 “\126\006\001\002\005\013\015”
#define OMP_O_MH_C_EXPANSION_RECORD     “\126\006\001\002\005\013\016”
#define OMP_O_MH_C_EXTENSIBLE_OBJECT    “\126\006\001\002\005\013\017”
#define OMP_O_MH_C_EXTENSION            “\126\006\001\002\005\013\020”
#define OMP_O_MH_C_EXTERNAL_TRACE_ENTRY “\126\006\001\002\005\013\021”
#define OMP_O_MH_C_G3_FAX_NBPS          “\126\006\001\002\005\013\022”
#define OMP_O_MH_C_GENERAL_CONTENT      “\126\006\001\002\005\013\023”
#define OMP_O_MH_C_INTERNAL_TRACE_ENTRY “\126\006\001\002\005\013\024”
#define OMP_O_MH_C_LOCAL_DELIV_CONFIRM  “\126\006\001\002\005\013\025”
#define OMP_O_MH_C_LOCAL_DELIV_CONFIRMS “\126\006\001\002\005\013\026”
#define OMP_O_MH_C_LOCAL_NDR            “\126\006\001\002\005\013\027”
#define OMP_O_MH_C_LOCAL_PER_RECIP_NDR  “\126\006\001\002\005\013\030”
#define OMP_O_MH_C_MESSAGE              “\126\006\001\002\005\013\031”
#define OMP_O_MH_C_MESSAGE_RD           “\126\006\001\002\005\013\032”
#define OMP_O_MH_C_MTS_IDENTIFIER       “\126\006\001\002\005\013\033”
#define OMP_O_MH_C_OR_ADDRESS           “\126\006\001\002\005\013\034”
#define OMP_O_MH_C_OR_NAME              “\126\006\001\002\005\013\035”
#define OMP_O_MH_C_PER_RECIP_DR         “\126\006\001\002\005\013\036”
#define OMP_O_MH_C_PER_RECIP_NDR        “\126\006\001\002\005\013\037”
#define OMP_O_MH_C_PER_RECIP_REPORT     “\126\006\001\002\005\013\040”
#define OMP_O_MH_C_PROBE                “\126\006\001\002\005\013\041”
#define OMP_O_MH_C_PROBE_RD             “\126\006\001\002\005\013\042”
#define OMP_O_MH_C_RD                   “\126\006\001\002\005\013\043”
#define OMP_O_MH_C_REDIRECTION_RECORD   “\126\006\001\002\005\013\044”
#define OMP_O_MH_C_REPORT               “\126\006\001\002\005\013\045”
#define OMP_O_MH_C_SECURITY_LABEL       “\126\006\001\002\005\013\046”
#define OMP_O_MH_C_SESSION              “\126\006\001\002\005\013\047”
```

```
#define OMP_O_MH_C_SUBMISSION_RESULTS   “\126\006\001\002\005\013\050”
#define OMP_O_MH_C_SUBMITTED_COMMUNIQUE “\126\006\001\002\005\013\051”
#define OMP_O_MH_C_SUBMITTED_MESSAGE    “\126\006\001\002\005\013\052”
#define OMP_O_MH_C_SUBMITTED_MESSAGE_RD “\126\006\001\002\005\013\053”
#define OMP_O_MH_C_SUBMITTED_PROBE      “\126\006\001\002\005\013\054”
#define OMP_O_MH_C_SUBMITTED_PROBE_RD   “\126\006\001\002\005\013\055”
#define OMP_O_MH_C_TELETEX_NBPS         “\126\006\001\002\005\013\056”
#define OMP_O_MH_C_DELIVERY_REPORT      “\126\006\001\002\005\013\057”
#define OMP_O_MH_C_MT_PUBLIC_DATA       “\126\006\001\002\005\013\060”
#define OMP_O_MH_C_TOKEN_PUBLIC_DATA    “\126\006\001\002\005\013\061”

#define OMP_O_MH_C_TOKEN                “\126\006\001\002\005\013\062”

/* 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)

        /* 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_MAX_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_LOSSESS      ((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_TELEX_TO_VIDEOTEX         ((OM_enumeration)   5)
#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_G4CLASS1      ((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_CLASS_1     ((OM_enumeration)  13)
#define MH_EC_TELETEX_TO_VIDEOTEX       ((OM_enumeration)  14)
#define MH_EC_VIDEOTEX_TO_TELEX         ((OM_enumeration)  15)
#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)

        /* RedirectionReason */
#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)

        /* Registration */
#define MH_RG_UNREGISTRED_MAIL          ((OM_enumeration)   0)
#define MH_RG_REGISTRED_MAIL            ((OM_enumeration)   1)
#define MH_RG_REGISTRED_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_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      “\126\003\003\001”
#define OMP_O_MH_CTO_UNIDENTIFIED       “\126\003\003\000”

        /* External EITs */
#define OMP_O_MH_EE_G3_FAX              “\126\003\004\003”
#define OMP_O_MH_EE_G4_CLASS_1          “\126\003\004\004”
#define OMP_O_MH_EE_IA5_TEXT            “\126\003\004\002”
#define OMP_O_MH_EE_MIXED_MODE          “\126\003\004\009”
#define OMP_O_MH_EE_TELETEX             “\126\003\004\005”
#define OMP_O_MH_EE_TELEX               “\126\003\004\001”
#define OMP_O_MH_EE_UNDEFINED           “\126\003\004\000”
#define OMP_O_MH_EE_VIDEOTEX            “\126\003\004\006”

        /* ODA EIT */
#define OMP_O_MH_ET_ODA_DATA            “\128\001\000\001”

        /* Rendition Attributes */
#define OMP_O_MH_RA_BASIC_RENDITION     “\126\003\005\000”

/* 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_INFORMATION      ((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_CONFIDENTIALITY_KEY        ((OM_type) 222)
#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_CONFIRMATIONS     ((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_MISCELLANEOUS_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_RECIPIENT_REPORTS      ((OM_type) 309)
#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_RANDOM_NUMBER              ((OM_type) 340)
#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_WORKSPACE                  ((OM_type) 376)
#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_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)   17)
#define MH_VL_LOCAL_IDENTIFIER         ((OM_value_length)   32)
#define MH_VL_MSG_CONTENT_CORRELATOR   ((OM_value_length)   16)
#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)
#define MH_VL_POSTAL_ADDRESS_IN_LINES  ((OM_value_length)   30)
```

```
/* Value Number */

#define MH_VN_BILATERAL_INFORMATION     ((OM_value_number)     8)
#define MH_VN_ENCODED_INFORMATION_TYPES ((OM_value_number)     8)
#define MH_VN_EXPANSION_HISTORY         ((OM_value_number)   512)
#define MH_VN_OTHER_RECIPIENT_NAMES     ((OM_value_number) 32767)
#define MH_VN_PREFERRED_DELIVERY_MODES  ((OM_value_number)    10)
#define MH_VN_RECIPIENT_DESCRIPTOR      ((OM_value_number) 32767)
#define MH_VN_REDIRECTION_HISTORY       ((OM_value_number)   512)
#define MH_VN_REPORT_SUBSTANCE          ((OM_value_number) 32767)
#define MH_VN_SECURITY_CATEGORY_DATA    ((OM_value_number)    64)
#define MH_VN_SECURITY_CATEGORY_IDS     ((OM_value_number)    64)
#define MH_VN_TRACE_INFO                ((OM_value_number)   512)

/* END MH PORTION OF INTERFACE */

#ifdef  __cplusplus
}
#endif

#endif  /* _XMHP_H */
```

A.4 ximp.h

The declarations contained in the header file ximp.h define the portion of the C constants definition that deals with IM packages. The symbols specified by these declarations are the only IM symbols that the service makes visible to the client. All applications that include this header must first include the OSI-Abstract-Data Manipulation header xom.h, the xmh.h header, and the xmhp.h header.

Note that 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 name are exceptions and are abbreviated as shown in Table A-2.

Table A-2 Abbreviations for ximp.h Class Names

Name	**Abbreviation**
NOTIFICATION	NOTIF
BODY_PART	BD_PRT
BILATERALLY	BILAT
EXTERNALLY	EXTERN
NATIONALLY	NATIONAL

```
#ident “@(#)ximp.h    1.3 22 Sep 1995 SMI”
/*
 *
 * Copyright 22 Sep 1995 Sun Microsystems, Inc. All Rights Reserved
 * (c) Copyright 1993 Marben Produit All rights reserved
 *
 */

#ifndef _XIMP_H
#define _XIMP_H

#ifdef  __cplusplus
extern “C” {
#endif
```

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

/* SYMBOLIC CONSTANTS */

/* Class */

#define OMP_O_IM_C_BILAT_DEF_BD_PRT     "\126\006\001\002\005\011\000"
#define OMP_O_IM_C_BD_PRT               "\126\006\001\002\005\011\001"
#define OMP_O_IM_C_EXTERN_DEF_BD_PRT    "\126\006\001\002\005\011\002"
#define OMP_O_IM_C_G3_FAX_BD_PRT        "\126\006\001\002\005\011\003"
#define OMP_O_IM_C_G4_CLASS_1_BD_PRT    "\126\006\001\002\005\011\004"
#define OMP_O_IM_C_IA5_TEXT_BD_PRT      "\126\006\001\002\005\011\005"
#define OMP_O_IM_C_INTERPERSONAL_MSG    "\126\006\001\002\005\011\006"
#define OMP_O_IM_C_INTERPERSONAL_NOTIF  "\126\006\001\002\005\011\007"
#define OMP_O_IM_C_IPM_IDENTIFIER       "\126\006\001\002\005\011\010"
#define OMP_O_IM_C_ISO_6937_TEXT_BD_PRT "\126\006\001\002\005\011\011"
#define OMP_O_IM_C_MESSAGE_BD_PRT       "\126\006\001\002\005\011\012"
#define OMP_O_IM_C_MIXED_MODE_BD_PRT    "\126\006\001\002\005\011\013"
#define OMP_O_IM_C_NATIONAL_DEF_BD_PRT  "\126\006\001\002\005\011\014"
#define OMP_O_IM_C_NON_RECEIPT_NOTIF    "\126\006\001\002\005\011\015"
#define OMP_O_IM_C_OR_DESCRIPTOR        "\126\006\001\002\005\011\016"
#define OMP_O_IM_C_RECEIPT_NOTIF        "\126\006\001\002\005\011\017"
#define OMP_O_IM_C_RECIPIENT_SPECIFIER  "\126\006\001\002\005\011\020"
#define OMP_O_IM_C_TELETEX_BD_PRT       "\126\006\001\002\005\011\021"
#define OMP_O_IM_C_VIDEOTEX_BD_PRT      "\126\006\001\002\005\011\022"
#define OMP_O_IM_C_ODA_BD_PRT           "\126\006\001\002\005\011\023"
#define OMP_O_IM_C_GENERAL_TEXT_BD_PRT  "\126\006\001\002\005\011\024"
#define OMP_O_IM_C_UNIDENTIFIED_BD_PRT  "\126\006\001\002\005\011\025"
#define OMP_O_IM_C_USA_NAT_DEF_BD_PRT   "\126\006\001\002\005\011\026"

/* 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)

        /* 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) 120)
#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)
/*Extra attribute type definition missing in the specification*/
#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)
#define IM_VL_LANGUAGES                 ((OM_value_length)  2)


/* END IM PORTION OF INTERFACE */

#ifdef  __cplusplus
}
#endif

#endif  /* _XIMP_H */
```

A.5 xedip.h

The declarations contained in the header file xedip.h define the portion of the C interface that deals with the EDI messaging package. The symbols specified by these declarations are the only EDI Messaging symbols that the service makes visible to the client. All applications that include this header must first include the OSI-Abstract-Data Manipulation header xom.h, the xmh.h header, and the xmhp.h header.

```
#ident “@(#)xedip.h   1.3 22 Sep 1995 SMI”
/*
 *
 * Copyright 22 Sep 1995 Sun Microsystems, Inc. All Rights Reserved
 * (c) Copyright 1993 Marben Produit All rights reserved
 *
 */

#ifndef _XEDIP_H
#define _XEDIP_H

#ifdef  __cplusplus
extern “C” {
#endif

/* BEGIN EDI PORTION OF INTERFACE */

/* SYMBOLIC CONSTANTS */

/* Class */

#define OMP_O_EDI_C_AUTHORIZATION_INFO        “\126\006\001\002\007\001\000”
#define OMP_O_EDI_C_CROSS_REF_INFO            “\126\006\001\002\007\001\001”
#define OMP_O_EDI_C_EDI_APP_SECUR_ELTS        “\126\006\001\002\007\001\002”
#define OMP_O_EDI_C_EDI_BODYPART              “\126\006\001\002\007\001\003”
#define OMP_O_EDI_C_EDI_EXTENSION             “\126\006\001\002\007\001\004”
#define OMP_O_EDI_C_EDI_HEADING               “\126\006\001\002\007\001\005”
#define OMP_O_EDI_C_EDI_NOTIF                 “\126\006\001\002\007\001\006”
#define OMP_O_EDI_C_EDI_NOTIF_REQUESTS        “\126\006\001\002\007\001\007”
#define OMP_O_EDI_C_EDIM                      “\126\006\001\002\007\001\010”
#define OMP_O_EDI_C_EDIM_BODYPART             “\126\006\001\002\007\001\011”
#define OMP_O_EDI_C_EDIM_EX_DF_BODYPART       “\126\006\001\002\007\001\012”
#define OMP_O_EDI_C_EDIM_ID                   “\126\006\001\002\007\001\013”
#define OMP_O_EDI_C_EDIN_RECEIVER             “\126\006\001\002\007\001\014”
#define OMP_O_EDI_C_FN_PDAU_REASON_CODE       “\126\006\001\002\007\001\015”
```

```
#define OMP_O_EDI_C_FN_REASON_CODE            “\126\006\001\002\007\001\016”
#define OMP_O_EDI_C_FN_UAMS_REASON_CODE       “\126\006\001\002\007\001\017”
#define OMP_O_EDI_C_FN_USER_REASON_CODE       “\126\006\001\002\007\001\020”
#define OMP_O_EDI_C_FORWARD_NOTIF             “\126\006\001\002\007\001\021”
#define OMP_O_EDI_C_INTER_SENDER_RECIP        “\126\006\001\002\007\001\022”
#define OMP_O_EDI_C_NEGATIVE_NOTIF            “\126\006\001\002\007\001\023”
#define OMP_O_EDI_C_NN_PDAU_REASON_CODE       “\126\006\001\002\007\001\024”
#define OMP_O_EDI_C_NN_REASON_CODE            “\126\006\001\002\007\001\025”
#define OMP_O_EDI_C_NN_UAMS_REASON_CODE       “\126\006\001\002\007\001\026”
#define OMP_O_EDI_C_NN_USER_REASON_CODE       “\126\006\001\002\007\001\027”
#define OMP_O_EDI_C_POSITIVE_NOTIF            “\126\006\001\002\007\001\030”
#define OMP_O_EDI_C_PRIMARY_BODYPART          “\126\006\001\002\007\001\031”
#define OMP_O_EDI_C_REASON_CODE               “\126\006\001\002\007\001\032”
#define OMP_O_EDI_C_RECIP_REFERENCE           “\126\006\001\002\007\001\033”
#define OMP_O_EDI_C_RECIPS                    “\126\006\001\002\007\001\034”
#define OMP_O_EDI_C_SECUR_ELEMENTS            “\126\006\001\002\007\001\035”
#define OMP_O_EDI_C_SERV_STRING_ADVICE        “\126\006\001\002\007\001\036”
#define OMP_O_EDI_C_SYNTAX_ID                 “\126\006\001\002\007\001\037”

/* Enumeration */

        /* EDI Initiator */
#define EDI_EI_INTERNAL_UA              ((OM_enumeration)   0)
#define EDI_EI_INTERNAL_MS              ((OM_enumeration)   1)
#define EDI_EI_EX_UA                    ((OM_enumeration)   2)

/* Integer */

        /* Content Type */
/* Note: this constant may be used as a value of the Content Type */
/*       attribute of the Communique class defined in the X.400 API */
#define EDI_CONTENT_TYPE                ((OM_integer)  35)

        /* FN PDAU Basic Code */
#define EDI_FPB_UNSPECIFIED             ((OM_integer)   0)
#define EDI_FPB_FORWARD_FOR_PB_AND_REND ((OM_integer)   1)

        /* FN UAMS Basic Code */
#define EDI_FUMB_UNSPECIFIED            ((OM_integer)   0)
#define EDI_FUMB_ONWARD_ROUTING         ((OM_integer)   1)
#define EDI_FUMB_RECIP_UNKNOWN          ((OM_integer)   2)
#define EDI_FUMB_ORIG_UNKNOWN           ((OM_integer)   3)
#define EDI_FUMB_FORWARD_BY_EDI_MS      ((OM_integer)   4)
```

```
        /* FN UAMS Diagnostic Code */
#define EDI_FUMD_RECIP_NAME_CHANGED     ((OM_integer)   1)
#define EDI_FUMD_RECIP_NAME_DELETED     ((OM_integer)   2)

        /* FN User Basic Code */
#define EDI_FUB_UNSPECIFIED             ((OM_integer)   0)
#define EDI_FUB_FORWARD_FOR_ARCHIVING   ((OM_integer)   1)
#define EDI_FUB_FORWARD_FOR_INFO        ((OM_integer)   2)
#define EDI_FUB_FORWARD_FOR_ADD_INFO    ((OM_integer)   3)
#define EDI_FUB_SUB_UNCHANGED           ((OM_integer)   4)
#define EDI_FUB_HEADING_FIELD_NOT_SUPP  ((OM_integer)   5)
#define EDI_FUB_BODYPART_TYPE_NOT_SUPP  ((OM_integer)   6)
#define EDI_FUB_MSG_TYPE_NOT_SUPP       ((OM_integer)   7)
#define EDI_FUB_SYNTAX_ID_NOT_SUPP      ((OM_integer)   8)
#define EDI_FUB_INTER_SENDER_UNKNOWN    ((OM_integer)   9)
#define EDI_FUB_USER_DEFINED_REASON     ((OM_integer)  10)

        /* NN PDAU Basic Code */
#define EDI_NPB_UNSPECIFIED             ((OM_integer)   0)
#define EDI_NPB_UNDELIV_MAIL            ((OM_integer)   1)
#define EDI_NPB_P_REND_NOT_PERFORMED    ((OM_integer)   2)

        /* NN NP Diagnostic Code */
#define EDI_NPD_P_REND_ATTS_NOT_SUPP    ((OM_integer)  31)
#define EDI_NPD_PD_ADDRESS_INCORRECT    ((OM_integer)  32)
#define EDI_NPD_PD_OFFICE_I_OR_INVALID  ((OM_integer)  33)
#define EDI_NPD_PD_ADDRESS_INC          ((OM_integer)  34)
#define EDI_NPD_RECIP_UNKNOWN           ((OM_integer)  35)
#define EDI_NPD_RECIP_DECEASED          ((OM_integer)  36)
#define EDI_NPD_ORGANIZATION_EXPIRED    ((OM_integer)  37)
#define EDI_NPD_RECIP_REFUSED_TO_ACCEPT ((OM_integer)  38)
#define EDI_NPD_RECIP_DID_NOT_CLAIM     ((OM_integer)  39)
#define EDI_NPD_RECIP_CHANGED_ADDRESS_P ((OM_integer)  40)
#define EDI_NPD_RECIP_CHANGED_ADDRESS_T ((OM_integer)  41)
#define EDI_NPD_RECIP_CHANGED_T_ADDRESS ((OM_integer)  42)
#define EDI_NPD_NEW_ADDRESS_UNKNOWN     ((OM_integer)  43)
#define EDI_NPD_RECIP_NOT_WANT_FORWARD  ((OM_integer)  44)
#define EDI_NPD_ORIG_PHOHIBITED_FORWARD ((OM_integer)  45)

        /* NN UAMS Basic Code */
#define EDI_NUMB_UNSPECIFIED            ((OM_integer)   0)
#define EDI_NUMB_CANNOT_DELIV_TO_USER   ((OM_integer)   1)
#define EDI_NUMB_DELIV_TIMEOUT          ((OM_integer)   2)
#define EDI_NUMB_MSG_DISCARDED          ((OM_integer)   3)
#define EDI_NUMB_SUB_TERMINATED         ((OM_integer)   4)
#define EDI_NUMB_FORWARDING_ERROR       ((OM_integer)   5)
```

```
#define EDI_NUMB_SECUR_ERROR            ((OM_integer)   6)

        /* NN NUM Diagnostic Code */
#define EDI_NUMD_PROTOCOL_VIOLATION     ((OM_integer)   1)
#define EDI_NUMD_EDIM_ORIG_UNKNOWN      ((OM_integer)   2)
#define EDI_NUMD_EDIM_RECIP_UNKNOWN     ((OM_integer)   3)
#define EDI_NUMD_EDIM_RECIP_AMBIGUOUS   ((OM_integer)   4)
#define EDI_NUMD_ACT_REQUEST_NOT_SUPP   ((OM_integer)   5)
#define EDI_NUMD_EDIM_EXPIRED           ((OM_integer)   6)
#define EDI_NUMD_EDIM_OBSOLETED         ((OM_integer)   7)
#define EDI_NUMD_DUPLICATE_EDIM         ((OM_integer)   8)
#define EDI_NUMD_UNSUPP_EXTENSION       ((OM_integer)   9)
#define EDI_NUMD_INC_COPY_REJECTED      ((OM_integer)  10)
#define EDI_NUMD_EDIM_TOO_LARGE_FOR_APP ((OM_integer)  11)
#define EDI_NUMD_FORWARD_EDIM_NOT_DELIV ((OM_integer)  12)
#define EDI_NUMD_FORWARD_EDIM_D_TIME    ((OM_integer)  13)
#define EDI_NUMD_FORWARD_LOOP_DETECTED  ((OM_integer)  14)
#define EDI_NUMD_UNABLE_TO_ACCEPT_RESP  ((OM_integer)  15)
#define EDI_NUMD_INTER_SENDER_UNKNOWN   ((OM_integer)  16)
#define EDI_NUMD_INTER_RECIP_UNKNOWN    ((OM_integer)  17)
#define EDI_NUMD_INVALID_HEADING_FIELD  ((OM_integer)  18)
#define EDI_NUMD_INVALID_BODYPART_TYPE  ((OM_integer)  19)
#define EDI_NUMD_INVALID_MSG_TYPE       ((OM_integer)  20)
#define EDI_NUMD_INVALID_SYNTAX_ID      ((OM_integer)  21)
#define EDI_NUMD_MSG_INTEG_FAIL         ((OM_integer)  22)
#define EDI_NUMD_FORWARD_MSG_INTEG_FAIL ((OM_integer)  23)
#define EDI_NUMD_UNSUPP_ALGORITHM       ((OM_integer)  24)
#define EDI_NUMD_DECRYPTION_FAILED      ((OM_integer)  25)
#define EDI_NUMD_TOKEN_ERROR            ((OM_integer)  26)
#define EDI_NUMD_UNABLE_TO_SIGN_NOTIF   ((OM_integer)  27)
#define EDI_NUMD_UNABLE_TO_MSG_R        ((OM_integer)  28)
#define EDI_NUMD_AUTHENTICATION_FAIL    ((OM_integer)  29)
#define EDI_NUMD_SECUR_CONTEXT_FAIL     ((OM_integer)  30)
#define EDI_NUMD_MSG_SEQUENCE_FAIL      ((OM_integer)  31)
#define EDI_NUMD_MSG_SECUR_LABEL_FAIL   ((OM_integer)  32)
#define EDI_NUMD_REPUDIATION_FAIL       ((OM_integer)  33)
#define EDI_NUMD_PROOF_OF_FAIL          ((OM_integer)  34)

        /* NN User Basic Code */
#define EDI_NUB_UNSPECIFIED             ((OM_integer)   0)
#define EDI_NUB_SYNTAX_ERROR            ((OM_integer)   1)
#define EDI_NUB_INTER_SENDER_UNKNOWN    ((OM_integer)   2)
#define EDI_NUB_INTER_RECIP_UNKNOWN     ((OM_integer)   3)
#define EDI_NUB_INVALID_HEADING_FIELD   ((OM_integer)   4)
#define EDI_NUB_INVALID_BOPYPART_TYPE   ((OM_integer)   5)
#define EDI_NUB_INVALID_MSG_TYPE        ((OM_integer)   6)
```

```
#define EDI_NUB_FUNCT_GROUP_NOT_SUPP    ((OM_integer)   7)
#define EDI_NUB_SUB_TERMINATED          ((OM_integer)   8)
#define EDI_NUB_NO_BILATERAL_AGREEMENT  ((OM_integer)   9)
#define EDI_NUB_USER_DEFINED_REASON     ((OM_integer)  10)


/* Object Identifier */

        /* EDI Package */
#define EDI_PACKAGE                     "\126\006\001\002\007\001"

        /* EDI Body Part Type Attribute */
#define OMP_O_EDI_BPO_EDIFACT_ISO646          "\126\006\007\013\000"
#define OMP_O_EDI_BPO_EDIFACT_T61             "\126\006\007\013\001"
#define OMP_O_EDI_BPO_EDIFACT_OCTET           "\126\006\007\013\002"
#define OMP_O_EDI_BPO_ANSIX12_ISO646          "\126\006\007\013\003"
#define OMP_O_EDI_BPO_ANSIX12_T61             "\126\006\007\013\004"
#define OMP_O_EDI_BPO_ANSIX12_OCTET           "\126\006\007\013\005"
#define OMP_O_EDI_BPO_ANSIX12_EBCDIC          "\126\006\007\013\006"
#define OMP_O_EDI_BPO_UNTDI_ISO646            "\126\006\007\013\007"
#define OMP_O_EDI_BPO_UNTDI_T61               "\126\006\007\013\010"
#define OMP_O_EDI_BPO_UNTDI_OCTET             "\126\006\007\013\011"
#define OMP_O_EDI_BPO_PRIVATE_OCTET           "\126\006\007\013\012"
#define OMP_O_EDI_BPO_UNDEFINED_OCTET         "\126\006\007\013\013"

        /* Action Request Attribute */
#define OMP_O_EDI_ARO_FOR_ACTION              "\126\006\007\015\000"
#define OMP_O_EDI_ARO_COPY                    "\126\006\007\015\001"

/* Type */

#define EDI_ACKNOWLEDGEMENT_REQUEST     ((OM_type) 1000)
#define EDI_ACT_REQUEST                 ((OM_type) 1001)
#define EDI_ADD_BODYPARTS               ((OM_type) 1002)
#define EDI_APP_CROSS_REFERENCE         ((OM_type) 1003)
#define EDI_APP_REFERENCE               ((OM_type) 1004)
#define EDI_APP_SECUR_ELEMENT           ((OM_type) 1005)
#define EDI_APP_SECUR_EXTENSIONS        ((OM_type) 1006)
#define EDI_AUTHORIZATION_INFO          ((OM_type) 1007)
#define EDI_AUTHORIZATION_INFO_Q        ((OM_type) 1008)
#define EDI_BASIC_CODE                  ((OM_type) 1009)
#define EDI_BODY                        ((OM_type) 1010)
#define EDI_BODYPART_REFERENCE          ((OM_type) 1011)
#define EDI_COMMS_AGREEMENT_ID          ((OM_type) 1012)
#define EDI_COMPONENT_DATA_ELEMENT_SEP  ((OM_type) 1013)
#define EDI_CRITICALITY                 ((OM_type) 1014)
```

```
#define EDI_CROSS_REFERENCING_INFO      ((OM_type) 1015)
#define EDI_DATA_ELEMENT_SEPARATOR      ((OM_type) 1016)
#define EDI_DATE_AND_TIME_OF_PREP       ((OM_type) 1017)
#define EDI_DECIMAL_NOTATION            ((OM_type) 1018)
#define EDI_DELIV_ENVELOPE              ((OM_type) 1019)
#define EDI_DIAGNOSTIC                  ((OM_type) 1020)
#define EDI_EDI_APP_SECUR_ELEMENTS      ((OM_type) 1021)
#define EDI_EDI_BODYPART_TYPE           ((OM_type) 1022)
#define EDI_EDI_MSG_TYPE                ((OM_type) 1023)
#define EDI_EDI_NOTIF_REQUESTS          ((OM_type) 1024)
#define EDI_EDI_SUPPLEMENTARY_INFO      ((OM_type) 1025)
#define EDI_EDIN_INITIATOR              ((OM_type) 1026)
#define EDI_EDIN_ORIG                   ((OM_type) 1027)
#define EDI_EDIN_RECEIVER               ((OM_type) 1028)
#define EDI_EDIN_RECEIVER_NAME          ((OM_type) 1029)
#define EDI_ENCRYPTED_PRIMARY_BODYPART  ((OM_type) 1030)
#define EDI_EXPIRY_TIME                 ((OM_type) 1031)
#define EDI_EXTENSION_TYPE              ((OM_type) 1032)
#define EDI_EXTENSION_VALUE             ((OM_type) 1033)
#define EDI_EX_BODYPART                 ((OM_type) 1034)
#define EDI_FIRST_RECIP                 ((OM_type) 1035)
#define EDI_FN_REASON_CODE              ((OM_type) 1036)
#define EDI_FN_SECUR_CHECK              ((OM_type) 1037)
#define EDI_FORWARD_NOTIF               ((OM_type) 1038)
#define EDI_FORWARD_TO                  ((OM_type) 1039)
#define EDI_HEADING                     ((OM_type) 1040)
#define EDI_HEADING_EXTENSIONS          ((OM_type) 1041)
#define EDI_IDENTIFICATION_CODE         ((OM_type) 1042)
#define EDI_IDENTIFICATION_CODE_Q       ((OM_type) 1043)
#define EDI_INC_COPY                    ((OM_type) 1044)
#define EDI_INTER_CONTROL_REFERENCE     ((OM_type) 1045)
#define EDI_INTER_RECIP                 ((OM_type) 1046)
#define EDI_INTER_SENDER                ((OM_type) 1047)
#define EDI_MSG_REFERENCE               ((OM_type) 1048)
#define EDI_NEGATIVE_NOTIF              ((OM_type) 1049)
#define EDI_NN_REASON_CODE              ((OM_type) 1050)
#define EDI_NON_REPUD_OF_NOTIF          ((OM_type) 1051)
#define EDI_NON_REPUD_OF_RECEPTION      ((OM_type) 1052)
#define EDI_NOTIF_EXTENSIONS            ((OM_type) 1053)
#define EDI_NOTIF_TIME                  ((OM_type) 1054)
#define EDI_OBSOLETED_EDIMS             ((OM_type) 1055)
#define EDI_ORIG                        ((OM_type) 1056)
#define EDI_ORIGINAL_CONTENT            ((OM_type) 1057)
#define EDI_ORIGINAL_CONTENT_INTEG_CK   ((OM_type) 1058)
#define EDI_ORIGINAL_EDIM_ID            ((OM_type) 1059)
#define EDI_POSITIVE_NOTIF              ((OM_type) 1060)
```

```
#define EDI_PRIMARY_BODYPART            ((OM_type) 1061)
#define EDI_PROCESSING_PRIORITY_CODE    ((OM_type) 1062)
#define EDI_PROOF_OF_NOTIFICATION       ((OM_type) 1063)
#define EDI_PROOF_OF_RECEPTION          ((OM_type) 1064)
#define EDI_RECIP                       ((OM_type) 1065)
#define EDI_RECIP_EXTENSIONS            ((OM_type) 1066)
#define EDI_RECIP_REFERENCE             ((OM_type) 1067)
#define EDI_RECIP_REFERENCE_Q           ((OM_type) 1068)
#define EDI_RECIPS                      ((OM_type) 1069)
#define EDI_RELATED_EDI_MSGS            ((OM_type) 1070)
#define EDI_RELATED_NON_EDI_MSGS        ((OM_type) 1071)
#define EDI_RELEASE_INDICATOR           ((OM_type) 1072)
#define EDI_RESERVED                    ((OM_type) 1073)
#define EDI_RESPONS_FORWARD             ((OM_type) 1074)
#define EDI_RESPONS_PASSING_ALLOWED     ((OM_type) 1075)
#define EDI_ROUTING_ADDRESS             ((OM_type) 1076)
#define EDI_SECUR_ELEMENTS              ((OM_type) 1077)
#define EDI_SEGMENT_TERMINATOR          ((OM_type) 1078)
#define EDI_SERVICE_STRING_ADVICE       ((OM_type) 1079)
#define EDI_SUBJECT_EDIM                ((OM_type) 1080)
#define EDI_SUPPLEMENTARY_INFO          ((OM_type) 1081)
#define EDI_SYNTAX_ID                   ((OM_type) 1082)
#define EDI_SYNTAX_ID_STRING            ((OM_type) 1083)
#define EDI_SYNTAX_VERSION              ((OM_type) 1084)
#define EDI_TEST_INDICATOR              ((OM_type) 1085)
#define EDI_THIS_EDIM                   ((OM_type) 1086)
#define EDI_USER                        ((OM_type) 1087)
#define EDI_USER_RELATIVE_ID            ((OM_type) 1088)

/* Value Length */

#define EDI_VL_APP_REFERENCE            ((OM_value_length)   14)
#define EDI_VL_APP_SECUR_ELEMENT        ((OM_value_length) 8191)
#define EDI_VL_AUTHORIZATION_INFO       ((OM_value_length)   10)
#define EDI_VL_AUTHORIZATION_INFO_Q     ((OM_value_length)    2)
#define EDI_VL_COMMS_AGREEMENT_ID       ((OM_value_length)   35)
#define EDI_VL_EDI_MSG_TYPE             ((OM_value_length)    6)
#define EDI_VL_IDENTIFICATION_CODE      ((OM_value_length)   35)
#define EDI_VL_IDENTIFICATION_CODE_Q    ((OM_value_length)    4)
#define EDI_VL_INTER_CONTROL_REFERENCE  ((OM_value_length)   14)
#define EDI_VL_RECIP_REFERENCE          ((OM_value_length)   14)
#define EDI_VL_RECIP_REFERENCE_Q        ((OM_value_length)    2)
#define EDI_VL_ROUTING_ADDRESS          ((OM_value_length)   14)
#define EDI_VL_SYNTAX_ID_STRING         ((OM_value_length)    4)
#define EDI_VL_SYNTAX_VERSION           ((OM_value_length)    5)
#define EDI_VL_USER_RELATIVE_ID         ((OM_value_length)   64)
```

```
/* END EDI PORTION OF INTERFACE */

#ifdef  __cplusplus
}
#endif

#endif  /* _XEDIP_H */
```

A.6 xms.h

The declarations contained in the header file xms.h define the portion of the C interface that deals with MS interface. The symbols specified by these declarations are the only MS symbols that the service makes visible to the client. All applications that include this header must first include the OSI-Abstract-Data Manipulation header xom.h.

```
#ident “@(#)xms.h      1.7 22 Sep 1995 SMI”
/*
 *
 * Copyright 22 Sep 1995 Sun Microsystems, Inc. All Rights Reserved
 * (c) Copyright 1993 Marben Produit All rights reserved
 *
 */

#ifndef _XMS_H
#define _XMS_H

#ifdef  __cplusplus
extern “C” {
#endif

/* BEGIN MS INTERFACE */

/* MS Package object identifier */

#define OMP_O_MS_PACKAGE                “\126\006\001\002\006\001”

/* Types Definitions */

typedef OM_private_object               MS_status;

typedef OM_sint                         MS_invoke_id;
typedef struct
{
    OM_object_identifier                feature;
    OM_boolean                          activated;
}
    MS_feature;

/* Interface Functions -- Function Prototypes */

MS_status ms_bind (
           OM_object          bind_argument ,
           OM_private_object  *bind_result_return ,
           OM_private_object  *bound_session_return
);

MS_status ms_cancel_submission (
           OM_private_object  session ,
           OM_object          mts_identifier ,
           MS_invoke_id       *invoke_id_return
);

MS_status ms_check_alert (
           OM_private_object  session ,
           OM_private_object  *check_alert_result_return
);

MS_status ms_delete (
           OM_private_object  session ,
           OM_object          delete_argument ,
           MS_invoke_id       *invoke_id_return
);

MS_status ms_fetch (
           OM_private_object  session ,
           OM_object          fetch_argument ,
           OM_private_object  *fetch_result_return ,
           MS_invoke_id       *invoke_id_return
);

MS_status ms_initialize (
           MS_feature         feature_list[] ,
           OM_sint            *max_sessions ,
           OM_sint            *max_outstanding_operations ,
           OM_workspace       *workspace_return
);
```

```
MS_status ms_list (
           OM_private_object  session ,
           OM_object          list_argument ,
           OM_private_object  *list_result_return ,
           MS_invoke_id       *invoke_id_return
);

MS_status ms_receive_result (
           OM_private_object  session ,
           OM_uint            *completion_flag_return ,
           MS_status          *operation_status_return ,
           OM_private_object  *result_return ,
           MS_invoke_id       *invoke_id_return
);

MS_status ms_register (
           OM_private_object  session ,
           OM_object          register_argument ,
           MS_invoke_id       *invoke_id_return
);

MS_status ms_register_ms (
           OM_private_object  session ,
           OM_object          register_ms_argument ,
           MS_invoke_id       *invoke_id_return
);

void  ms_shutdown (
           void
);

MS_status ms_submit (
           OM_private_object  session ,
           OM_object          communique ,
           OM_private_object  *submission_results_return ,
           MS_invoke_id       *invoke_id_return
);

MS_status ms_summarize (
           OM_private_object  session ,
           OM_object          summarize_argument ,
           OM_private_object  *summarize_result_return ,
           MS_invoke_id       *invoke_id_return
);
```

```
MS_status ms_unbind (
           OM_private_object  session
);

MS_status ms_wait (
           OM_private_object  session ,
           OM_uint32          interval ,
           OM_private_object  *wait_result_return ,
           MS_invoke_id       *invoke_id_return
);

/* SYMBOLIC CONSTANTS */

/* Classes */

/*
 * Note: Every client program must explicitly import into
 * every compilation unit (C source program) the classes or
 * Object Identifiers that it uses. Each of theses classes or
 * Object Identifier names must the be explicitly exported from
 * just one compilation unit.
 * Importing and exporting can be done using the OM_IMPORT and
 * OM_EXPORT macros respectively (see [OM API]).
 * For instance, the client program uses
 *          OM_IMPORTS( MS_C_ATTRIBUTE_DEFAULTS )
 * which in turn will make use of
 *          OMP_O_MS_C_ATTRIBUTE_DEFAULTS
 * defined below.
*/

#define OMP_O_MS_C_ATTRIBUTE_DEFAULTS         “\126\006\001\002\006\001\001”
#define OMP_O_MS_C_ATTRIBUTE_ERROR            “\126\006\001\002\006\001\002”
#define OMP_O_MS_C_ATTRIBUTE_PROBLEM          “\126\006\001\002\006\001\003”
#define OMP_O_MS_C_ATTRIBUTE_SELECTION        “\126\006\001\002\006\001\004”
#define OMP_O_MS_C_AUTO_ACTION                “\126\006\001\002\006\001\005”
#define OMP_O_MS_C_AUTO_ACTION_DEREGISTRATION “\126\006\001\002\006\001\006”
#define OMP_O_MS_C_AUTO_ACTION_REGISTRATION   “\126\006\001\002\006\001\007”
#define OMP_O_MS_C_AUTO_ACTION_REQUEST_ERROR  “\126\006\001\002\006\001\010”
#define OMP_O_MS_C_AUTO_ACTION_REQUESTPROBLEM “\126\006\001\002\006\001\011”
#define OMP_O_MS_C_BIND_ARGUMENT              “\126\006\001\002\006\001\012”
#define OMP_O_MS_C_BIND_ERROR                 “\126\006\001\002\006\001\013”
#define OMP_O_MS_C_BIND_RESULT                “\126\006\001\002\006\001\014”
#define OMP_O_MS_C_CANCEL_SUBMISSION_ERROR    “\126\006\001\002\006\001\015”
#define OMP_O_MS_C_CHANGE_CREDENTIALS         “\126\006\001\002\006\001\016”
#define OMP_O_MS_C_CHECK_ALERT_RESULT         “\126\006\001\002\006\001\017”
#define OMP_O_MS_C_COMMON_CONTROLS            “\126\006\001\002\006\001\020”
```

```
#define OMP_O_MS_C_COMMUNICATIONS_ERROR       “\126\006\001\002\006\001\021”
#define OMP_O_MS_C_CREATION_TIME_RANGE        “\126\006\001\002\006\001\022”
#define OMP_O_MS_C_CREDENTIALS                “\126\006\001\002\006\001\023”
#define OMP_O_MS_C_DEFAULT_DELIVERY_CONTROLS  “\126\006\001\002\006\001\024”
#define OMP_O_MS_C_DELETE_ARGUMENT            “\126\006\001\002\006\001\025”
#define OMP_O_MS_C_DELETE_ERROR               “\126\006\001\002\006\001\026”
#define OMP_O_MS_C_DELETE_PROBLEM             “\126\006\001\002\006\001\027”
#define OMP_O_MS_C_DELIVERABLE_CONTENT_TYPES  “\126\006\001\002\006\001\030”
#define OMP_O_MS_C_ELMT_NOT_SUBSCRIBED_ERROR  “\126\006\001\002\006\001\031”
#define OMP_O_MS_C_ERROR                      “\126\006\001\002\006\001\032”
#define OMP_O_MS_C_FETCH_ARGUMENT             “\126\006\001\002\006\001\033”
#define OMP_O_MS_C_FETCH_ATTRIBUTE_DEFAULTS   “\126\006\001\002\006\001\034”
#define OMP_O_MS_C_FETCH_RESTRICTION_ERROR    “\126\006\001\002\006\001\035”
#define OMP_O_MS_C_FETCH_RESTRICTION_PROBLEM  “\126\006\001\002\006\001\036”
#define OMP_O_MS_C_FETCH_RESULT               “\126\006\001\002\006\001\037”
#define OMP_O_MS_C_INCONSISTENT_REQUEST_ERROR “\126\006\001\002\006\001\040”
#define OMP_O_MS_C_INVALID_PARAMETERS_ERROR   “\126\006\001\002\006\001\041”
#define OMP_O_MS_C_ITEM                       “\126\006\001\002\006\001\042”
#define OMP_O_MS_C_ITEM_TO_FORWARD            “\126\006\001\002\006\001\043”
#define OMP_O_MS_C_ITEMS                      “\126\006\001\002\006\001\044”
#define OMP_O_MS_C_LABEL_AND_REDIRECTION      “\126\006\001\002\006\001\045”
#define OMP_O_MS_C_LABELS_AND_REDIRECTIONS    “\126\006\001\002\006\001\046”
#define OMP_O_MS_C_LIBRARY_ERROR              “\126\006\001\002\006\001\047”
#define OMP_O_MS_C_LIST_ARGUMENT              “\126\006\001\002\006\001\050”
#define OMP_O_MS_C_LIST_ATTRIBUTE_DEFAULTS    “\126\006\001\002\006\001\051”
#define OMP_O_MS_C_LIST_RESULT                “\126\006\001\002\006\001\052”
#define OMP_O_MS_C_MS_ENTRY_INFO_SELECTION    “\126\006\001\002\006\001\053”
#define OMP_O_MS_C_MS_ENTRY_INFORMATION       “\126\006\001\002\006\001\054”
#define OMP_O_MS_C_ORIGINATOR_INVALID_ERROR   “\126\006\001\002\006\001\055”
#define OMP_O_MS_C_PASSWORD                   “\126\006\001\002\006\001\056”
#define OMP_O_MS_C_RANGE                      “\126\006\001\002\006\001\057”
#define OMP_O_MS_C_RANGE_ERROR                “\126\006\001\002\006\001\060”
#define OMP_O_MS_C_RECIPIENT_IMPROPER_ERROR   “\126\006\001\002\006\001\061”
#define OMP_O_MS_C_REGISTER_ARGUMENT          “\126\006\001\002\006\001\062”
#define OMP_O_MS_C_REGISTER_MS_ARGUMENT       “\126\006\001\002\006\001\063”
#define OMP_O_MS_C_REGISTER_REJECTED_ERROR    “\126\006\001\002\006\001\064”
#define OMP_O_MS_C_REMOTE_BIND_ERROR          “\126\006\001\002\006\001\065”
#define OMP_O_MS_C_RESTRICTIONS               “\126\006\001\002\006\001\066”
#define OMP_O_MS_C_SECURITY_ERROR             “\126\006\001\002\006\001\067”
#define OMP_O_MS_C_SELECTOR                   “\126\006\001\002\006\001\070”
#define OMP_O_MS_C_SEQUENCE_NUMBER_ERROR      “\126\006\001\002\006\001\071”
#define OMP_O_MS_C_SEQUENCE_NUMBER_PROBLEM    “\126\006\001\002\006\001\072”
#define OMP_O_MS_C_SEQUENCE_NUMBER_RANGE      “\126\006\001\002\006\001\073”
#define OMP_O_MS_C_SERVICE_ERROR              “\126\006\001\002\006\001\074”
#define OMP_O_MS_C_SESSION                    “\126\006\001\002\006\001\075”
#define OMP_O_MS_C_STRONG_CREDENTIALS         “\126\006\001\002\006\001\076”
```

```
#define OMP_O_MS_C_SUBMIT_CTRL_VIOLATED_ERROR “\126\006\001\002\006\001\077”
#define OMP_O_MS_C_SUMMARIZE_ARGUMENT         “\126\006\001\002\006\001\100”
#define OMP_O_MS_C_SUMMARY                    “\126\006\001\002\006\001\101”
#define OMP_O_MS_C_SUMMARY_PRESENT            “\126\006\001\002\006\001\102”
#define OMP_O_MS_C_SUMMARY_REQUESTS           “\126\006\001\002\006\001\103”
#define OMP_O_MS_C_SUMMARY_RESULT             “\126\006\001\002\006\001\104”
#define OMP_O_MS_C_SYSTEM_ERROR               “\126\006\001\002\006\001\105”
#define OMP_O_MS_C_UNSUPT_CRITICAL_FUNC_ERROR “\126\006\001\002\006\001\106”
#define OMP_O_MS_C_WAIT_RESULT                “\126\006\001\002\006\001\107”
/*
** WARNING : the following constants have been added in order to implement
             SNI-2 X/OPEN CR
             (Date: February 1992 Ref: XoTGnet 3619
*/
#define OMP_O_MS_C_ALERT_ADDRESS              “\126\006\001\002\006\001\110”
#define OMP_O_MS_C_AUTO_ALERT_REG_PARAMETER   “\126\006\001\002\006\001\111”
#define OMP_O_MS_C_AUTO_FORWARD_REG_PARAMETER “\126\006\001\002\006\001\112”
#define OMP_O_MS_C_AUTO_FORWARD_ARGUMENTS     “\126\006\001\002\006\001\113”

/*
** WARNING : the following constant have been added in order to implement
             the MTS Change Credentials operation
*/
#define OMP_O_MS_C_CHANGE_CRED_ARGUMENT       “\126\006\001\002\006\001\114”

/* OM Attribute Names */

#define MS_ABSENT                          ((OM_type) 1201)
#define MS_ADDRESS                         ((OM_type) 1202)
#define MS_ALERT_INDICATION                ((OM_type) 1203)
#define MS_ALERT_REGISTRATION_IDENTIFIER ((OM_type) 1204)
#define MS_ALLOWED_CONTENT_TYPES           ((OM_type) 1205)
#define MS_ALLOWED_EITS                    ((OM_type) 1206)
#define MS_ALTERNATE_RECIPIENT             ((OM_type) 1207)
#define MS_ATTRIBUTE_TYPE                  ((OM_type) 1208)
#define MS_ATTRIBUTE_PROBLEMS              ((OM_type) 1209)
#define MS_ATTRIBUTE_VALUE                 ((OM_type) 1210)
#define MS_ATTRIBUTES                      ((OM_type) 1211)
#define MS_AUTO_ACTION_DEREGISTRATIONS     ((OM_type) 1212)
#define MS_AUTO_ACTION_REGISTRATIONS       ((OM_type) 1213)
#define MS_AUTO_ACTION_REQUEST_PROBLEMS    ((OM_type) 1214)
#define MS_AVA                             ((OM_type) 1215)
#define MS_AVAILABLE_ATTRIBUTE_TYPES       ((OM_type) 1216)
#define MS_AVAILABLE_AUTO_ACTIONS          ((OM_type) 1217)
#define MS_BIND_TOKEN                      ((OM_type) 1218)
#define MS_CERTIFICATE                     ((OM_type) 1219)
```

```
#define MS_CHANGE_CREDENTIALS           ((OM_type) 1220)
#define MS_CHILD_ENTRIES                ((OM_type) 1221)
#define MS_CONTENT_LENGTH               ((OM_type) 1222)
#define MS_CONTENT_TYPE                 ((OM_type) 1223)
#define MS_CONTENT_TYPE_INT             ((OM_type) 1224)
#define MS_CONTENT_TYPES_SUPPORTED      ((OM_type) 1225)
#define MS_COUNT                        ((OM_type) 1226)
#define MS_CREATION_TIME_RANGE          ((OM_type) 1227)
#define MS_DEFAULT_DELIVERY_CONTROLS    ((OM_type) 1228)
#define MS_DELETE_PROBLEMS              ((OM_type) 1229)
#define MS_DELIVERABLE_CONTENT_TYPES    ((OM_type) 1230)
#define MS_DELIVERABLE_EIT              ((OM_type) 1231)
#define MS_DELIVERABLE_MAX_CONTENT_LEN  ((OM_type) 1232)
#define MS_EITS                         ((OM_type) 1233)
#define MS_ENTRY_INFORMATION            ((OM_type) 1234)
#define MS_FETCH_ATTRIBUTE_DEFAULTS     ((OM_type) 1235)
#define MS_FETCH_RESTRICTION_PROBLEMS   ((OM_type) 1236)
#define MS_FETCH_RESTRICTIONS           ((OM_type) 1237)
#define MS_FILE_DESCRIPTOR              ((OM_type) 1238)
#define MS_FILTER                       ((OM_type) 1239)
#define MS_FROM                         ((OM_type) 1240)
#define MS_FROM_INT                     ((OM_type) 1241)
#define MS_IA5_STRING                   ((OM_type) 1242)
#define MS_INFORMATION_BASE_TYPE        ((OM_type) 1243)
#define MS_INITIATOR                    ((OM_type) 1244)
#define MS_INITIATOR_CREDENTIALS        ((OM_type) 1245)
#define MS_ITEM                         ((OM_type) 1246)
#define MS_ITEMS                        ((OM_type) 1247)
#define MS_LABEL_AND_REDIRECTION        ((OM_type) 1248)
#define MS_LABELS_AND_REDIRECTIONS      ((OM_type) 1249)
#define MS_LIMIT                        ((OM_type) 1250)
#define MS_LIST                         ((OM_type) 1251)
#define MS_LIST_ATTRIBUTE_DEFAULTS      ((OM_type) 1252)
#define MS_MAX_CONTENT_LENGTH           ((OM_type) 1253)
#define MS_MS_CONFIGURATION_REQUEST     ((OM_type) 1254)
#define MS_NAME                         ((OM_type) 1255)
#define MS_NEW_CREDENTIALS              ((OM_type) 1256)
#define MS_NEW_ENTRY                    ((OM_type) 1257)
#define MS_NEXT                         ((OM_type) 1258)
#define MS_OCTET_STRING                 ((OM_type) 1259)
#define MS_OLD_CREDENTIALS              ((OM_type) 1260)
#define MS_OVERRIDE                     ((OM_type) 1261)
#define MS_PERMISSIBLE_CONTENT_TYPES    ((OM_type) 1262)
#define MS_PERMISSIBLE_EITS             ((OM_type) 1263)
#define MS_PERMISSIBLE_LOWEST_PRIORITY  ((OM_type) 1264)
#define MS_PERMISSIBLE_MAX_CONTENT_LEN  ((OM_type) 1265)
```

```
#define MS_PERMISSIBLE_OPERATIONS        ((OM_type) 1266)
#define MS_PRECISE                       ((OM_type) 1267)
#define MS_PRESENT                       ((OM_type) 1268)
#define MS_PROBLEM                       ((OM_type) 1269)
#define MS_RANGE                         ((OM_type) 1270)
#define MS_RECIPIENTS                    ((OM_type) 1271)
#define MS_REGISTRATION_ID               ((OM_type) 1272)
#define MS_REGISTRATION_PARAMETER        ((OM_type) 1273)
#define MS_REQUESTED                     ((OM_type) 1274)
#define MS_REQUESTED_ATTRIBUTES          ((OM_type) 1275)
#define MS_REQUESTED_STR                 ((OM_type) 1276)
#define MS_RESPONDER_CREDENTIALS         ((OM_type) 1277)
#define MS_RESTRICT                      ((OM_type) 1278)
#define MS_SECURITY_CONTEXT              ((OM_type) 1279)
#define MS_SECURITY_LABEL                ((OM_type) 1280)
#define MS_SECURITY_PROBLEM              ((OM_type) 1281)
#define MS_SELECTION                     ((OM_type) 1282)
#define MS_SELECTOR                      ((OM_type) 1283)
#define MS_SEQUENCE_NUMBER               ((OM_type) 1284)
#define MS_SEQUENCE_NUMBER_PROBLEMS      ((OM_type) 1285)
#define MS_SEQUENCE_NUMBER_RANGE         ((OM_type) 1286)
#define MS_SIMPLE                        ((OM_type) 1287)
#define MS_SPAN                          ((OM_type) 1288)
#define MS_STRONG                        ((OM_type) 1289)
#define MS_SUMMARIES                     ((OM_type) 1290)
#define MS_SUMMARY_REQUESTS              ((OM_type) 1291)
#define MS_TO                            ((OM_type) 1292)
#define MS_TO_INT                        ((OM_type) 1293)
#define MS_TYPE                          ((OM_type) 1294)
#define MS_USER_SECURITY_LABELS          ((OM_type) 1295)
#define MS_WAIT_NEW_AVAILABLE            ((OM_type) 1296)
/*
** WARNING : the following constants have been added because
             of a lack in the X/OPEN MS API Prelim Spec
             (Date: January 1992 Ref: XO/PRELIM/91/060
              ISBN: 1 872630 26 X)
*/
#define MS_PERMISSIBLE_CONTENT_TYPES_INT ((OM_type) 1297)
/*
** WARNING : the following constants have been added in order to implement
             SNI-2 X/OPEN CR
             (Date: February 1992 Ref: XoTGnet 3619
*/
#define MS_AUTO_FORWARD_ARGUMENTS        ((OM_type) 1298)
#define MS_DELETE_AFTER_AUTO_FORWARD     ((OM_type) 1299)
#define MS_OTHER_PARAMETERS              ((OM_type) 1300)
```

```
#define MS_PER_RECIPIENT_FIELDS          ((OM_type) 1301)
#define MS_ALERT_QUALIFIER               ((OM_type) 1302)
#define MS_ALERT_ADDRESS                 ((OM_type) 1303)

/* Enumeration */
/*
 * The following enumerations tags and enumeration constants
 * are defined for use as values of the corresponding OM attributes.
*/

/* Enumeration Tags for MS_Problem: */

#define MS_E_ACTION_TYPE_NOT_SUBSCRIBED     ((OM_enumeration) 1)
#define MS_E_ASYNCHRONY_NOT_SUPPORTED       ((OM_enumeration) 2)
#define MS_E_ATTRIB_TYPE_NOT_SUBSCRIBED     ((OM_enumeration) 3)
#define MS_E_AUTHENTICATION_ERROR           ((OM_enumeration) 4)
#define MS_E_BAD_ARGUMENT                   ((OM_enumeration) 5)
#define MS_E_BAD_CLASS                      ((OM_enumeration) 6)
#define MS_E_BAD_SESSION                    ((OM_enumeration) 7)
#define MS_E_BUSY                           ((OM_enumeration) 8)
#define MS_E_CHILD_ENTRY_SPECIFIED          ((OM_enumeration) 9)
#define MS_E_COMMUNICATIONS_PROBLEM         ((OM_enumeration) 10)
#define MS_E_CONTENT_LENGTH_PROBLEM         ((OM_enumeration) 11)
#define MS_E_CONTENT_TYPE_PROBLEM           ((OM_enumeration) 12)
#define MS_E_DEFER_DELIV_CANCEL_REJECT      ((OM_enumeration) 13)
#define MS_E_DELETE_RESTRICTION_PROBLEM     ((OM_enumeration) 14)
#define MS_E_EIT_PROBLEM                    ((OM_enumeration) 15)
#define MS_E_ELMT_SERV_NOT_SUBSCRIBED       ((OM_enumeration) 16)
#define MS_E_FEATURE_NOT_NEGOTIATED         ((OM_enumeration) 17)
#define MS_E_FEATURE_UNAVAILABLE            ((OM_enumeration) 18)
#define MS_E_INAPPROPRIATE_FOR_OPERATN      ((OM_enumeration) 19)
#define MS_E_INAPPROPRIATE_MATCHING         ((OM_enumeration) 20)
#define MS_E_INCONSISTENT_REQUEST           ((OM_enumeration) 21)
#define MS_E_INVALID_ATTRIBUTE_VALUE        ((OM_enumeration) 22)
#define MS_E_INVALID_FEATURE                ((OM_enumeration) 23)
#define MS_E_INVALID_PARAMETERS             ((OM_enumeration) 24)
#define MS_E_MESSAGE_SUBMIT_ID_INVALID      ((OM_enumeration) 25)
#define MS_E_MISCELLANEOUS                  ((OM_enumeration) 26)
#define MS_E_NO_SUCH_CLASS                  ((OM_enumeration) 27)
#define MS_E_NO_SUCH_ENTRY                  ((OM_enumeration) 28)
#define MS_E_ORIGINATOR_INVALID             ((OM_enumeration) 29)
#define MS_E_OUT_OF_MEMORY                  ((OM_enumeration) 30)
#define MS_E_RECIPIENT_IMPROPERLY_SPECIFIED ((OM_enumeration) 31)
#define MS_E_REGISTER_REJECTED              ((OM_enumeration) 32)
#define MS_E_REMOTE_BIND_ERROR              ((OM_enumeration) 33)
#define MS_E_REVERSED                       ((OM_enumeration) 34)
```

```
#define MS_E_SECURITY                         ((OM_enumeration) 35)
#define MS_E_SUBMISSION_CTRL_VIOLATED         ((OM_enumeration) 36)
#define MS_E_TOO_MANY_OPERATIONS              ((OM_enumeration) 37)
#define MS_E_UNABLE_ESTABLISH_ASSOCIATN       ((OM_enumeration) 38)
#define MS_E_UNACCEPTABLE_SECURE_CONTXT       ((OM_enumeration) 39)
#define MS_E_UNAVAILABLE                      ((OM_enumeration) 40)
#define MS_E_UNAVAILABLE_ACTION_TYPE          ((OM_enumeration) 41)
#define MS_E_UNAVAILABLE_ATTRIBUTE_TYPE       ((OM_enumeration) 42)
#define MS_E_UNSUPPORTED_CRITICAL_FUNC        ((OM_enumeration) 43)
#define MS_E_UNWILLING_TO_PERFORM             ((OM_enumeration) 44)

/*
** WARNING : the following constants have been added because
             of a lack in the X/OPEN MS API Prelim Spec
             (Date: January 1992 Ref: XO/PRELIM/91/060
              ISBN: 1 872630 26 X)
*/
#define MS_E_TOO_MANY_SESSIONS                ((OM_enumeration) 45)

/* Constants */

#define MS_DEFAULT_FEATURE_LIST           ((MS_feature) 0)
#define MS_SUCCESS                        ((MS_status) 0)
#define MS_NO_WORKSPACE                   ((MS_status) 1)

/* Constants of type OM_object */

#define MS_NO_FILTER                      ((OM_object) 0)
#define MS_NO_NEW_ENTRIES                 ((OM_object) 0)
#define MS_NULL_RESULT                    ((OM_object) 0)
#define MS_OPERATION_NOT_STARTED          ((OM_object) 0)

/* Constants of type Integer */

/*    Completion-Flag (Unsigned-Integer): */

#define MS_COMPLETED_OPERATION            ((OM_uint) 1)
#define MS_OUTSTANDING_OPERATION          ((OM_uint) 2)
#define MS_NO_OUTSTANDING_OPERATION       ((OM_uint) 3)

/*    Information-Base-Type (Integer): */

#define MS_STORED_MESSAGES                ((OM_integer) 0)
#define MS_INLOG                          ((OM_integer) 1)
#define MS_OUTLOG                         ((OM_integer) 2)
```

```
/* Feature */
#define OMP_O_MS_FE                        “\126\006\001\002\006\004”
#define OMP_O_MS_FE_SUBMISSION             “\126\006\001\002\006\005”
#define OMP_O_MS_FE_ADMINISTRATION         “\126\006\001\002\006\006”
#define OMP_O_MS_FE_ALERT                  “\126\006\001\002\006\007”

#ifdef  __cplusplus
}
#endif

#endif /* _XMS_H */
```

A.7 xmsga.h

The declarations contained in the header file `xmsga.h` defines the MS General Attributes Package. All applications that include this header must first include the OSI-Abstract-Data Manipulation header `xom.h` and the `xms.h` header.

```
#ident “@(#)xmsga.h   1.5 22 Sep 1995 SMI”
/*
 *
 * Copyright 22 Sep 1995 Sun Microsystems, Inc. All Rights Reserved
 * (c) Copyright 1993 Marben Produit All rights reserved
 *
 */

#ifndef _XMSGA_H
#define _XMSGA_H

#ifdef  __cplusplus
extern “C” {
#endif

/* BEGIN MS INTERFACE */

/* MS General Attributes Package object identifier */

#define OMP_O_MS_GENERAL_ATTRIBUTES_PACKAGE “\126\006\001\002\006\002”

/* SYMBOLIC CONSTANTS */
/* MS General Attributes Types */
/*
 * Note: Every client program must explicitly import into
 * every compilation unit (C source program) the classes or
 * Object Identifiers that it uses. Each of theses classes or
 * Object Identifier names must the be explicitly exported from
 * just one compilation unit.
 * Importing and exporting can be done using the OM_IMPORT and
 * OM_EXPORT macros respectively (see [OM API]).
 * For instance, the client program uses
 *          OM_IMPORTS( MS_A_CHILD_SEQUENCE_NUMBERS )
 * which in turn will make use of
 *          OMP_O_MS_A_SEQUENCE_NUMBERS
 * defined below.
*/
#define OMP_O_MS_A_CHILD_SEQUENCE_NUMBERS     “\126\004\003\000”
#define OMP_O_MS_A_CONTENT                    “\126\004\003\001”
#define OMP_O_MS_A_CONTENT_CONFIDENTL_ALGM_ID “\126\004\003\002”
#define OMP_O_MS_A_CONTENT_CORRELATOR         “\126\004\003\003”
#define OMP_O_MS_A_CONTENT_IDENTIFIER         “\126\004\003\004”
#define OMP_O_MS_A_CONTENT_INTEGRITY_CHECK    “\126\004\003\005”
#define OMP_O_MS_A_CONTENT_LENGTH             “\126\004\003\006”
#define OMP_O_MS_A_CONTENT_RETURNED           “\126\004\003\007”
#define OMP_O_MS_A_CONTENT_TYPE               “\126\004\003\010”
#define OMP_O_MS_A_CONVERSION_LOSS_PROHIBITED “\126\004\003\011”
#define OMP_O_MS_A_CONVERTED_EITS             “\126\004\003\012”
#define OMP_O_MS_A_CREATION_TIME              “\126\004\003\013”
#define OMP_O_MS_A_DELIVERED_EITS             “\126\004\003\014”
#define OMP_O_MS_A_DELIVERY_FLAGS             “\126\004\003\015”
#define OMP_O_MS_A_DL_EXPANSION_HISTORY       “\126\004\003\016”
#define OMP_O_MS_A_ENTRY_STATUS               “\126\004\003\017”
#define OMP_O_MS_A_ENTRY_TYPE                 “\126\004\003\020”
#define OMP_O_MS_A_INTENDED_RECIPIENT_NAME    “\126\004\003\021”
#define OMP_O_MS_A_MESSAGE_DELIVERY_ENVELOPE  “\126\004\003\022”
#define OMP_O_MS_A_MESSAGE_DELIVERY_ID        “\126\004\003\023”
#define OMP_O_MS_A_MESSAGE_DELIVERY_TIME      “\126\004\003\024”
#define OMP_O_MS_A_MESSAGE_ORIGIN_AUTHEN_CHK  “\126\004\003\025”
#define OMP_O_MS_A_MESSAGE_SECURITY_LABEL     “\126\004\003\026”
#define OMP_O_MS_A_MESSAGE_SUBMISSION_TIME    “\126\004\003\027”
#define OMP_O_MS_A_MESSAGE_TOKEN              “\126\004\003\030”
#define OMP_O_MS_A_ORIGINAL_EITS              “\126\004\003\031”
#define OMP_O_MS_A_ORIGINATOR_CERTIFICATE     “\126\004\003\032”
#define OMP_O_MS_A_ORIGINATOR_NAME            “\126\004\003\033”
#define OMP_O_MS_A_OTHER_RECIPIENT_NAMES      “\126\004\003\034”
#define OMP_O_MS_A_PARENT_SEQUENCE_NUMBER     “\126\004\003\035”
#define OMP_O_MS_A_PERRECIP_REPORT_DELIV_FLDS “\126\004\003\036”
```

```
#define OMP_O_MS_A_PRIORITY                      “\126\004\003\037”
#define OMP_O_MS_A_PROOF_OF_DELIVERY_REQUEST  “\126\004\003\040”
#define OMP_O_MS_A_REDIRECTION_HISTORY        “\126\004\003\041”
#define OMP_O_MS_A_REPORT_DELIVERY_ENVELOPE   “\126\004\003\042”
#define OMP_O_MS_A_REPORTING_DL_NAME          “\126\004\003\043”
#define OMP_O_MS_A_REPORTING_MTA_CERTIFICATE  “\126\004\003\044”
#define OMP_O_MS_A_REPORT_ORIGIN_AUTHEN_CHK   “\126\004\003\045”
#define OMP_O_MS_A_SECURITY_CLASSIFICATION    “\126\004\003\046”
#define OMP_O_MS_A_SEQUENCE_NUMBER            “\126\004\003\047”
#define OMP_O_MS_A_SUBJECT_SUBMISSION_ID      “\126\004\003\050”
#define OMP_O_MS_A_THIS_RECIPIENT_NAME        “\126\004\003\051”

/* Enumeration Constants */

/*    for MS_A_ENTRY_STATUS */

#define MS_ES_NEW                         ((OM_enumeration) 0)
#define MS_ES_LISTED                      ((OM_enumeration) 1)
#define MS_ES_PROCESSED                   ((OM_enumeration) 2)

/*    for MS_A_ENTRY_TYPE */

#define MS_ET_DELIVERED_MESSAGE           ((OM_enumeration) 0)
#define MS_ET_DELIVERED_REPORT            ((OM_enumeration) 1)
#define MS_ET_RETURNED_CONTENT            ((OM_enumeration) 2)

/*    for MS_A_PRIORITY */

#define MS_PTY_NORMAL                     ((OM_enumeration) 0)
#define MS_PTY_LOW                        ((OM_enumeration) 1)
#define MS_PTY_URGENT                     ((OM_enumeration) 2)

/*    for MS_A_SECURITY_CLASSIFICATION */

#define MS_SC_UNMARKED                    ((OM_enumeration) 0)
#define MS_SC_UNCLASSIFIED                ((OM_enumeration) 1)
#define MS_SC_RESTRICTED                  ((OM_enumeration) 2)
#define MS_SC_CONFIDENTIAL                ((OM_enumeration) 3)
#define MS_SC_SECRET                      ((OM_enumeration) 4)
#define MS_SC_TOP_SECRET                  ((OM_enumeration) 5)

#ifdef  __cplusplus
}
#endif

#endif /* _XMSGA_H */
```

A.8 xmsima.h

The declarations contained in the header file `xmsima.h` defines the MS Interpersonal Messaging Package. All applications that include this header must first include the OSI-Abstract-Data Manipulation header `xom.h`, the `xms.h` header, and the `xmsga.h` header.

```
#ident “@(#)xmsima.h   1.6 22 Sep 1995 SMI”
/*
 * Copyright 22 Sep 1995 Sun Microsystems, Inc. All Rights Reserved
 * (c) Copyright 1993 Marben Produit All rights reserved
 */

#ifndef _XMSIMA_H
#define _XMSIMA_H

#ifdef  __cplusplus
extern “C” {
#endif

/* MS Interpersonal Messaging Attributes Package object identifier */

#define OMP_O_MS_IM_ATTRIBUTES_PACKAGE “\126\006\001\002\006\003”

/* SYMBOLIC CONSTANTS */

/* MS Interpersonal Messaging Attributes Types */
/*
 * Note: Every client program must explicitly import into
 * every compilation unit (C source program) the classes or
 * Object Identifiers that it uses. Each of theses classes or
 * Object Identifier names must the be explicitly exported from
 * just one compilation unit.
 * Importing and exporting can be done using the OM_IMPORT and
 * OM_EXPORT macros respectively (see [OM API]).
 * For instance, the client program uses
 *         OM_IMPORTS( MS_IM_A_ACKNOWLEDGMENT_MODE )
 * which in turn will make use of
 *         OMP_O_MS_IM_A_ACKNOWLEDGMENT_MODE
 * defined below.
*/
```

```
#define OMP_O_MS_IM_A_ACKNOWLEDGMENT_MODE       “\126\001\011\011”
#define OMP_O_MS_IM_A_AUTHORIZING_USERS         “\126\001\007\012”
#define OMP_O_MS_IM_A_AUTO_FORWARD_COMMENT      “\126\001\011\006”
#define OMP_O_MS_IM_A_AUTO_FORWARDED            “\126\001\007\011”
#define OMP_O_MS_IM_A_BILATERAL_DEF_BODY_PARTS  “\126\001\010\012”
#define OMP_O_MS_IM_A_BLIND_COPY_RECIPIENTS     “\126\001\007\015”
#define OMP_O_MS_IM_A_BODY                      “\126\001\010\000”
#define OMP_O_MS_IM_A_CONVERSION_EITS           “\126\001\011\003”
#define OMP_O_MS_IM_A_COPY_RECIPIENTS           “\126\001\007\014”
#define OMP_O_MS_IM_A_DISCARD_REASON            “\126\001\011\005”
#define OMP_O_MS_IM_A_EXPIRY_TIME               “\126\001\007\005”
#define OMP_O_MS_IM_A_EXTENDED_BODY_PART_TYPES  “\126\001\010\014”
#define OMP_O_MS_IM_A_G3_FAX_BODY_PARTS         “\126\001\010\003”
#define OMP_O_MS_IM_A_G3_FAX_DATA               “\126\001\010\026”
#define OMP_O_MS_IM_A_G3_FAX_PARAMETERS         “\126\001\010\017”
#define OMP_O_MS_IM_A_G4_CLASS1_BODY_PARTS      “\126\001\010\004”
#define OMP_O_MS_IM_A_HEADING                   “\126\001\007\000”
#define OMP_O_MS_IM_A_IA5_TEXT_BODY_PARTS       “\126\001\010\001”
#define OMP_O_MS_IM_A_IA5_TEXT_DATA             “\126\001\010\024”
#define OMP_O_MS_IM_A_IA5_TEXT_PARAMETERS       “\126\001\010\015”
#define OMP_O_MS_IM_A_IMPORTANCE                “\126\001\007\007”
#define OMP_O_MS_IM_A_INCOMPLETE_COPY           “\126\001\007\021”
#define OMP_O_MS_IM_A_IPM_ENTRY_TYPE            “\126\001\006\000”
#define OMP_O_MS_IM_A_IPM_PREFERRED_RECIPIENT   “\126\001\011\002”
#define OMP_O_MS_IM_A_IPM_SYNOPSIS              “\126\001\006\001”
#define OMP_O_MS_IM_A_IPN_ORIGINATOR            “\126\001\011\001”
#define OMP_O_MS_IM_A_LANGUAGES                 “\126\001\007\022”
#define OMP_O_MS_IM_A_MESSAGE_BODY_PARTS        “\126\001\010\010”
#define OMP_O_MS_IM_A_MESSAGE_DATA              “\126\001\010\032”
#define OMP_O_MS_IM_A_MESSAGE_PARAMETERS        “\126\001\010\023”
#define OMP_O_MS_IM_A_MIXED_MODE_BODY_PARTS     “\126\001\010\011”
#define OMP_O_MS_IM_A_NATIONAL_DEF_BODY_PARTS   “\126\001\010\013”
#define OMP_O_MS_IM_A_NON_RECEIPT_REASON        “\126\001\011\004”
#define OMP_O_MS_IM_A_NRN_REQUESTORS            “\126\001\007\024”
#define OMP_O_MS_IM_A_OBSOLETED_IPMS            “\126\001\007\016”
#define OMP_O_MS_IM_A_ORIGINATOR                “\126\001\007\002”
#define OMP_O_MS_IM_A_PRIMARY_RECIPIENTS        “\126\001\007\013”
#define OMP_O_MS_IM_A_RECEIPT_TIME              “\126\001\011\010”
#define OMP_O_MS_IM_A_RELATED_IPMS              “\126\001\007\017”
#define OMP_O_MS_IM_A_REPLIED_TO_IPM            “\126\001\007\003”
#define OMP_O_MS_IM_A_REPLY_RECIPIENTS          “\126\001\007\020”
#define OMP_O_MS_IM_A_REPLY_REQUESTORS          “\126\001\007\025”
#define OMP_O_MS_IM_A_REPLY_TIME                “\126\001\007\006”
#define OMP_O_MS_IM_A_RETURNED_IPM              “\126\001\011\007”
#define OMP_O_MS_IM_A_RN_REQUESTORS             “\126\001\007\023”
#define OMP_O_MS_IM_A_SENSITIVITY               “\126\001\007\010”
```

```
#define OMP_O_MS_IM_A_SUBJECT                  “\126\001\007\004”
#define OMP_O_MS_IM_A_SUBJECT_IPM              “\126\001\011\000”
#define OMP_O_MS_IM_A_SUPPL_RECEIPT_INFO       “\126\001\011\012”
#define OMP_O_MS_IM_A_TELETEX_BODY_PARTS       “\126\001\010\005”
#define OMP_O_MS_IM_A_TELETEX_DATA             “\126\001\010\027”
#define OMP_O_MS_IM_A_TELETEX_PARAMETERS       “\126\001\010\020”
#define OMP_O_MS_IM_A_THIS_IPM                 “\126\001\007\001”
#define OMP_O_MS_IM_A_VIDEOTEX_BODY_PARTS      “\126\001\010\006”
#define OMP_O_MS_IM_A_VIDEOTEX_DATA            “\126\001\010\030”
#define OMP_O_MS_IM_A_VIDEOTEX_PARAMETERS      “\126\001\010\021”

/* Classes in the MS IM Attributes Package */

#define OMP_O_MS_IM_C_BODYPART_SYNOPSIS        “\126\006\001\002\006\003\001”
#define OMP_O_MS_IM_C_HEADING                  “\126\006\001\002\006\003\002”
#define OMP_O_MS_IM_C_IPM_SYNOPSIS             “\126\006\001\002\006\003\003”
#define OMP_O_MS_IM_C_MSG_BODYPART_SYNOPSIS    “\126\006\001\002\006\003\004”
#define OMP_O_MS_IM_C_NON_MSG_BODYPART_SYNOPSIS “\126\006\001\002\006\003\005”
#define OMP_O_MS_IM_C_TELETEX_PARAMETERS       “\126\006\001\002\006\003\006”
/*
** WARNING : all following classes have been added by Marben Produit in order
**           to support Body, G3Fax Data MS Attributes
**           because X/OPEN definition
**           is quiet incorrect.
*/
#define OMP_O_MS_IM_C_BODY                     “\126\006\001\002\006\003\007”
#define OMP_O_MS_IM_C_G3FAX_DATA               “\126\006\001\002\006\003\010”
#define OMP_O_MS_IM_C_TELETEX_DATA             “\126\006\001\002\006\003\011”

#define OMP_O_MS_CTO_P2_1984          “\126\001\012\000”
#define OMP_O_MS_CTO_P2_1988          “\126\001\012\001”


/* OM Attributes Names in the MS IM Attributes Package */

#define MS_IM_AUTHORIZING_USERS               ((OM_type) 1401)
#define MS_IM_AUTO_FORWARDED                  ((OM_type) 1402)
#define MS_IM_BLIND_COPY_RECIPIENTS           ((OM_type) 1403)
#define MS_IM_BODY_PART_SYNOPSIS              ((OM_type) 1404)
#define MS_IM_COPY_RECIPIENTS                 ((OM_type) 1405)
#define MS_IM_EXPIRY_TIME                     ((OM_type) 1406)
#define MS_IM_EXTENSIONS                      ((OM_type) 1407)
#define MS_IM_IMPORTANCE                      ((OM_type) 1408)
#define MS_IM_MESSAGE_BODY_PART_SYNOPSIS      ((OM_type) 1409)
#define MS_IM_NON_MESSAGE_BODY_PART_SYNOPSIS ((OM_type) 1410)
#define MS_IM_NUMBER_OF_PAGES                 ((OM_type) 1411)
```

```
#define MS_IM_OBSOLETED_IPMS                ((OM_type) 1412)
#define MS_IM_ORIGINATOR                    ((OM_type) 1413)
#define MS_IM_PARAMETERS                    ((OM_type) 1414)
#define MS_IM_PRIMARY_RECIPIENTS            ((OM_type) 1415)
#define MS_IM_PROCESSED                     ((OM_type) 1416)
#define MS_IM_RELATED_IPMS                  ((OM_type) 1417)
#define MS_IM_REPLIED_TO_IPM                ((OM_type) 1418)
#define MS_IM_REPLY_RECIPIENTS              ((OM_type) 1419)
#define MS_IM_REPLY_TIME                    ((OM_type) 1420)
#define MS_IM_SENSITIVITY                   ((OM_type) 1421)
#define MS_IM_SEQUENCE_NUMBER               ((OM_type) 1422)
#define MS_IM_SIZE                          ((OM_type) 1423)
#define MS_IM_SUBJECT                       ((OM_type) 1424)
#define MS_IM_SYNOPSIS                      ((OM_type) 1425)
#define MS_IM_THIS_IPM                      ((OM_type) 1426)
#define MS_IM_TYPE                          ((OM_type) 1427)
/*
** WARNING : all following attributes have been added by
**          Marben Produit in order
**          to support Body, G3Fax Data MS
**          Attributes because X/OPEN definition
**          is quiet incorrect.
*/
#define MS_IM_BODY_PART                     ((OM_type) 1428)
#define MS_IM_IMAGES                        ((OM_type) 1429)
#define MS_IM_TELETEX_DOCUMENT              ((OM_type) 1430)

/* Enumeration Constants */

/*    for MS_IM_ACKOWLEDGMENT_MODE */

#define MS_AM_AUTOMATIC                ((OM_enumeration) 0)
#define MS_AM_MANUAL                   ((OM_enumeration) 1)

/*    for MS_IM_DISCARD_REASON */

#define MS_DR_NO_DISCARD               ((OM_enumeration) -1)
#define MS_DR_IPM_EXPIRED              ((OM_enumeration) 0)
#define MS_DR_IPM_OBSOLETED            ((OM_enumeration) 1)
#define MS_DR_USER_TERMINATED          ((OM_enumeration) 2)

/*    for MS_IM_IA5_REPERTOIRE */

#define MS_IR_IA5                      ((OM_enumeration) 2)
#define MS_IR_ITA2                     ((OM_enumeration) 5)
```

```
/*    for MS_IM_IMPORTANCE */

#define MS_IM_LOW                       ((OM_enumeration) 0)
#define MS_IM_ROUTINE                   ((OM_enumeration) 1)
#define MS_IM_HIGH                      ((OM_enumeration) 2)

/*    for MS_IM_IPM_ENTRY_TYPE */

#define MS_IE_IPM                       ((OM_enumeration) 0)
#define MS_IE_RN                        ((OM_enumeration) 1)
#define MS_IE_NRN                       ((OM_enumeration) 2)

/*    for MS_IM_NR_REASON */

#define MS_NR_IPM_AUTO_FORWARDED        ((OM_enumeration) 0)
#define MS_NR_IPM_DISCARDED             ((OM_enumeration) 1)

/*    for MS_IM_SENSITIVITY */

#define MS_SE_NOT_SENSITIVE             ((OM_enumeration) 0)
#define MS_SE_PERSONAL                  ((OM_enumeration) 1)
#define MS_SE_PRIVATE                   ((OM_enumeration) 2)
#define MS_SE_COMPANY_CONFIDENTIAL      ((OM_enumeration) 3)

/*    for MS_IM_VIDEOTEX_SYNTAX */

#define MS_VS_UNSPECIFIED               ((OM_enumeration) -1)
#define MS_VS_IDS                       ((OM_enumeration) 0)
#define MS_VS_DATA_SYNTAX_1             ((OM_enumeration) 1)
#define MS_VS_DATA_SYNTAX_2             ((OM_enumeration) 2)
#define MS_VS_DATA_SYNTAX_3             ((OM_enumeration) 3)

#ifdef  __cplusplus
}
#endif

#endif /* _XMSIMA_H */
```

A.9 xmsedia.h

The declarations contained in the header file xmsima.h defines the MS EDI Package. All applications that include this header must first include the OSI-Abstract-Data Manipulation header xom.h, the xms.h header, and the xmsga.h header.

```
#ident “@(#)xmsedia.h 1.3 22 Sep 1995 SMI”
/*
 *
 * Copyright 22 Sep 1995 Sun Microsystems, Inc. All Rights Reserved
 * (c) Copyright 1993 Marben Produit All rights reserved
 *
 */

#ifndef _XMSEDIA_H
#define _XMSEDIA_H

#ifdef  __cplusplus
extern “C” {
#endif

/* MS Edi Attributes Package object identifier */

#define OMP_O_MS_EDI_ATTRIBUTES_PACKAGE “\126\006\001\002\007\002”

/* SYMBOLIC CONSTANTS */

/* MS Edi Attributes Types */
/*
 * Note: Every client program must explicitly import into
 * every compilation unit (C source program) the classes or
 * Object Identifiers that it uses. Each of theses classes or
 * Object Identifier names must the be explicitly exported from
 * just one compilation unit.
 * Importing and exporting can be done using the OM_IMPORT and
 * OM_EXPORT macros respectively (see [OM API]).
 * For instance, the client program uses
 *         OM_IMPORTS( MS_EDI_A_ACK_REQUEST )
 * which in turn will make use of
 *         OMP_O_MS_EDI_A_ACK_REQUEST
 * defined below.
*/
```

```
#define OMP_O_MS_EDI_A_ACK_REQUEST            “\126\007\007\010”
#define OMP_O_MS_EDI_A_ACTION_REQUEST         “\126\007\007\001”
#define OMP_O_MS_EDI_A_APPLICATION_REF        “\126\007\006\023”
#define OMP_O_MS_EDI_A_AUTH_INFO              “\126\007\007\016”
#define OMP_O_MS_EDI_A_BODY                   “\126\007\010\000”
#define OMP_O_MS_EDI_A_COMMS_AGREE_ID         “\126\007\007\011”
#define OMP_O_MS_EDI_A_CROSS_REF_INFO         “\126\007\006\015”
#define OMP_O_MS_EDI_A_DATE_TIME_PREP         “\126\007\006\022”
#define OMP_O_MS_EDI_A_EDI_AP_SECUR_ELT       “\126\007\006\013”
#define OMP_O_MS_EDI_A_EDI_AP_SECUR_EXTS      “\126\007\006\014”
#define OMP_O_MS_EDI_A_EDI_BD_PRT             “\126\007\010\002”
#define OMP_O_MS_EDI_A_EDI_BD_PRT_TYPE        “\126\007\006\006”
#define OMP_O_MS_EDI_A_EDI_MESSAGE_TYPE       “\126\007\006\016”
#define OMP_O_MS_EDI_A_EDI_NOTIF_IND          “\126\007\005\002”
#define OMP_O_MS_EDI_A_EDI_NOTIF_REQS         “\126\007\007\002”
#define OMP_O_MS_EDI_A_EDI_NOTIF_SECUR        “\126\007\007\013”
#define OMP_O_MS_EDI_A_EDI_RECEP_SECUR        “\126\007\007\014”
#define OMP_O_MS_EDI_A_EDIM_BD_PRT            “\126\007\010\003”
#define OMP_O_MS_EDI_A_EDIM_SYNOPSIS          “\126\007\005\001”
#define OMP_O_MS_EDI_A_EDIMS_ENTRY_TYPE       “\126\007\005\000”
#define OMP_O_MS_EDI_A_EDIN_INITIATOR         “\126\007\011\006”
#define OMP_O_MS_EDI_A_EDIN_ORIGINATOR        “\126\007\011\001”
#define OMP_O_MS_EDI_A_EDIN_RECEIVER          “\126\007\006\004”
#define OMP_O_MS_EDI_A_EXPIRY_TIME            “\126\007\006\010”
#define OMP_O_MS_EDI_A_EX_DF_BD_PRT_TYPE      “\126\007\010\006”
#define OMP_O_MS_EDI_A_FIRST_RECIP            “\126\007\011\002”
#define OMP_O_MS_EDI_A_FN_EXTENSIONS          “\126\007\011\017”
#define OMP_O_MS_EDI_A_FN_REASON_CODE         “\126\007\011\015”
#define OMP_O_MS_EDI_A_FN_SUPP_INFO           “\126\007\011\016”
#define OMP_O_MS_EDI_A_FORWARDED_TO           “\126\007\011\014”
#define OMP_O_MS_EDI_A_HEADING                “\126\007\006\000”
#define OMP_O_MS_EDI_A_HEADING_EXTENSION      “\126\007\006\024”
#define OMP_O_MS_EDI_A_INCOMPLETE_COPY        “\126\007\006\007”
#define OMP_O_MS_EDI_A_INTER_CONTROL_REF      “\126\007\007\006”
#define OMP_O_MS_EDI_A_INTER_LENGTH           “\126\007\010\001”
#define OMP_O_MS_EDI_A_INTER_RECIP            “\126\007\007\004”
#define OMP_O_MS_EDI_A_INTER_SENDER           “\126\007\006\021”
#define OMP_O_MS_EDI_A_MSG_DATA               “\126\007\010\005”
#define OMP_O_MS_EDI_A_MSG_PARAMETERS         “\126\007\010\004”
#define OMP_O_MS_EDI_A_NN_EXTENSIONS          “\126\007\011\013”
#define OMP_O_MS_EDI_A_NN_REASON_CODE         “\126\007\011\011”
#define OMP_O_MS_EDI_A_NN_SUPP_INFO           “\126\007\011\012”
#define OMP_O_MS_EDI_A_NOTIF_SECUR_ELTS       “\126\007\011\004”
#define OMP_O_MS_EDI_A_NOTIF_TIME             “\126\007\011\003”
#define OMP_O_MS_EDI_A_NOTIF_EXTENSIONS       “\126\007\011\005”
#define OMP_O_MS_EDI_A_OBSOLETED_EDIMS        “\126\007\006\012”
```

```
#define OMP_O_MS_EDI_A_ORIGINATOR             “\126\007\006\002”
#define OMP_O_MS_EDI_A_PN_EXTENSIONS          “\126\007\011\010”
#define OMP_O_MS_EDI_A_PN_SUPP_INFO           “\126\007\011\007”
#define OMP_O_MS_EDI_A_PROC_PRIORITY          “\126\007\007\007”
#define OMP_O_MS_EDI_A_RECIP_EXTENSIONS       “\126\007\007\015”
#define OMP_O_MS_EDI_A_RECIP_REF              “\126\007\007\005”
#define OMP_O_MS_EDI_A_RELATED_MESSAGES       “\126\007\006\011”
#define OMP_O_MS_EDI_A_RESPONS_FORWARDED      “\126\007\006\005”
#define OMP_O_MS_EDI_A_RESPONS_PASS_ALL       “\126\007\007\003”
#define OMP_O_MS_EDI_A_SERV_STR_ADVICE        “\126\007\006\017”
#define OMP_O_MS_EDI_A_SUBJECT_EDIM           “\126\007\011\000”
#define OMP_O_MS_EDI_A_SYNTAX_ID              “\126\007\006\020”
#define OMP_O_MS_EDI_A_TEST_INDICATOR         “\126\007\007\012”
#define OMP_O_MS_EDI_A_THIS_EDIM              “\126\007\006\001”
#define OMP_O_MS_EDI_A_THIS_RECIP             “\126\007\007\000”

/* Classes in the MS EDI Attributes Package */

#define OMP_O_MS_EDI_C_EDI_BODY               “\126\006\001\002\007\002\001”

/* OM Attributes Names in the MS EDI Attributes Package */

#define MS_EDI_PRIMARY_BODYPART              ((OM_type) 1501)
#define MS_EDI_ADD_BODYPARTS                 ((OM_type) 1502)

/* Enumeration Constants */

/*    for MS_EDI_NOTIF_IND */

#define MS_NI_NOTIFICATION_SENT         ((OM_enumeration) 0)
#define MS_NI_PN_SENT                   ((OM_enumeration) 1)
#define MS_NI_NN_SENT                   ((OM_enumeration) 2)
#define MS_NI_FN_SENT                   ((OM_enumeration) 3)

/*    for MS_EDI_EDIMS_ENTRY_TYPE */

#define MS_EE_EDIM                      ((OM_enumeration) 0)
#define MS_EE_PN                        ((OM_enumeration) 1)
#define MS_EE_NN                        ((OM_enumeration) 2)
#define MS_EE_FN                        ((OM_enumeration) 3)

#ifdef  __cplusplus
}
#endif

#endif /* _XMSEDIA_H */
```

MA and MT Interface Return Codes B

This appendix contains details of the return codes for the Message Access (MA) and Message Transfer (MT) interfaces.

Note that these are the return codes indicated in the error section of each of the function definitions in Chapter 2, “Message Access (MA) Interface” and Chapter 3, “Message Transfer (MT) Interface”.

MS interface errors are represented by private objects. Details are in Chapter 9, “Message Store Package Errors”.

B.1 Return Code Descriptions

The return codes are summarized in Table B-1.

Table B-1 Return Code Descriptions.

Return Code	Description
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 recognize the specified client name.
no-such-client-instance	The service does not recognize 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 recognize 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 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.

B.2 MA Interface Return Codes

Table B-2 lists the MA interface specific return codes and indicates to which MA interface functions they are applicable. Return codes that are also used by the OM interface are indicated by italics.

Table B-2 MA Interface Return Codes

Return Code	**Can**	**Clo**	**FiD**	**Ope**	**Siz**	**StD**	**Sub**	**Wai**
bad-message	-	-	-	-	-	x	-	-
feature-conflicts	-	-	-	x	-	-	-	-
feature-unavailable	x	-	x	x	x	x	x	x
function-interrupted	x	x	x	x	x	x	x	x
memory-insufficient	x	x	x	x	x	x	x	x
network-error	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	-
no-such-session	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
not-reserved	-	-	-	-	-	-	-	-
not-unreserved	-	-	-	-	-	-	-	-
originator-improper	-	-	-	-	-	-	x	-
permanent-error	x	x	x	x	x	x	x	x
pointer-invalid	x	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	x
system-error	x	x	x	x	x	x	x	x
temporary-error	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
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	-

B.3 MT Interface Return Codes

Table B-3 lists the MT interface specific return codes and indicates to which MT interface functions they are applicable. Return codes that are also used by the OM interface are indicated by italics.

Table B-3 MT 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	-

Referenced Documents C

This appendix identifies other documents referenced in, or related to, this document.

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 *Solstice XOM Programming Reference Manual* provided by SunSoft is derived from this X/Open text.

The following documents constitute X.400 (1984). Developed under the auspices of, and ratified by, the International Telecommunication Union (ITU; formerly known as the 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, ITU (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.
- *ISI/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 ITU, 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, MT and MS interfaces and packages.

- *Recommendation X.400, Message Handling: System and Service Overview, ITU (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 10021-2)
- *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 5*, Source: ITU (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, 22 February 1991.

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

- *Recommendation F.1, Operational Provisions for the International Public Telegram Service, ITU (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.

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

The following document defines the X.500 Directory API (XDS). It defines the Certificates, Name and Presentation Address classes upon which the MH and MS packages depend.

- *XDS - API to Directory Services, CAE Specification*, X/Open Company Limited in conjunction with X.400 API Association, 1991. The *Solstice X.500 XDS Programming Reference Manual* (part number 802-2130) provided by SunSoft is derived from this X/Open text.

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 ITU, it provides postal delivery mode definitions for the MH packages.

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

The following documents apply to the EDI Messaging Package specified in Chapter 7, "EDI Messaging Package".

- *Message Handling: EDI Messaging System*, ITU (CCITT) Recommendation X.435 | ISO / IEC 10021-9
- *Message Handling: EDI Messaging Service*, ITU (CCITT) Recommendation F.435 | ISO / IEC 10021-8
- *Electronic Data Interchange for Administration, Commerce and Transport (EDIFACT)*, ISO 9735, 1987
- *United Nations Trade Data Interchange (UN/TDI)*, United Nations Economic Commission for Europe, 1985
- *American National Standards Institute X12* (ANSIX12)

Glossary

1988 class	A class all the attributes specific to which are for X.400 (1988) and not X.400 (1984).
alert operation	An MS abstract operation which allows the MS to signal, based on selection criteria, to the User Agent that messages or reports are waiting in the MS. This operation can only be issued on an existing session (or association) with the MS.
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.
auto action	Actions that can be performed automatically by the MS, based on previously registered information from the MS-owner via the User Agent.
auto action type	An auto action type is used to indicate the type of auto action, e.g., Alert.
auto alert	An auto action, within the MS, triggered by delivery of a message and which, based on the registered criteria, may cause an alert to be generated.

autoforward

The automatic action within the MS which triggers a message to be forwarded automatically to another recipient (or other recipients) by the MS. The message may optionally be deleted.

C interface

A version of an interface for the variant of C standardized by ANSI.

child entry

An entry, other than the main-entry in an MS information base. The parent-entry for a child-entry can be either the main-entry or another child-entry depending on the number of levels in each case.

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.

delete operation

An MS abstract operation used to delete one or more entries from an information base.

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.

fetch operation

An MS abstract operation which allows one entry or parts of an entry to be fetched from the stored messages information base.

forwarding request

A parameter that may be present in a Message Submission operation, invoked by the User Agent, to request that a message be forwarded from the MS.

functional unit

A group of related functions.

generic interface	A version of an interface that is independent of any particular programming language.
information base	Objects within the MS which store information relevant to the MS abstract service, e.g., the stored messages information base, which stores the messages and reports that have been delivered into the MS.
input queue	The database by means of which the service conveys objects to the client of the MT interface.
list operation	An MS abstract operation which allows a selection of entries from an information base and requested attribute information to be returned for those entries.
local environment	The environment of the local MTA.
local MD	The management domain (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).
MS interface	The X.400 Message Store API.
MT interface	The X.400 Gateway API.
neighboring 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.
organizational unit name attributes	The Organizational Unit Name 1, Organizational Unit Name 2, Organizational Unit Name 3 and Organizational 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.
register-MS operation	An MS operation that allows the User Agent to register certain information, that is relevant to interworking between the User Agent and the MS, within the MS.
registration	Information registered in the MS and stored (until changed by the *Register-MS Operation*) between sessions (or associations).
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 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.

stored messages	The type of information base used to store entries containing messages and reports delivered by the Message Transfer System to the MS.
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.
summarize operation	An MS operation that allows a quick overview of the kind and number of entries which are currently stored in an information base.
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.

Index