

INTERNATIONAL STANDARD

ISO/IEC
14361

First edition
1996-06-01

Information technology — MHS-based electronic messaging — Application Program Interface (API) [Language independent]

*Technologies de l'information — Messagerie électronique basée sur
MHS — Interface de programme d'application (API) [indépendante du
langage]*



Reference number
ISO/IEC 14361:1996(E)



Documents

Document Sharing Hub

Contents

	PAGE
Section 1: General	1
1.1 Scope	1
1.2 Normative References	2
1.3 Conformance	5
1.4 Test Methods	9
Section 2: Terminology and General Requirements	11
2.1 Conventions	11
2.2 Definitions	14
Section 3: Overview	25
3.1 Purpose	25
3.2 Overview	26
3.3 Object Management	26
3.3.1 Syntax	26
3.3.2 Value	27
3.3.3 OM Attribute	27
3.3.4 OM Object	27
3.3.5 OM Class	27
3.3.6 Package	29
3.3.7 Package Closure	29
3.3.8 Workspace	29
3.3.9 Use of Objects	30
3.4 Features	30
Section 4: Message Handling Interfaces	33
4.1 Datatypes	33
4.1.1 Boolean List	33
4.1.2 Feature List	34
4.1.3 Interval	34
4.1.4 Object Count	34
4.1.5 Sequence Number	34
4.2 Access Operations	34
4.2.1 Cancel Submission	37
4.2.2 Close	37

© ISO/IEC 1996

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from the publisher.

ISO/IEC Copyright Office • Case postale 56 • CH-1211 Genève 20 • Switzerland

Printed in Switzerland

4.2.3	Finish Delivery	38
4.2.4	Finish Retrieval	39
4.2.5	Open	40
4.2.6	Size	42
4.2.7	Start Delivery	43
4.2.8	Start Retrieval	44
4.2.9	Submit	45
4.2.10	Wait	46
4.3	Transfer Operations	48
4.3.1	Close	50
4.3.2	Finish Transfer In	50
4.3.3	Open	52
4.3.4	Size	53
4.3.5	Start Transfer In	54
4.3.6	Transfer Out	55
4.3.7	Wait	56
4.4	Return Codes	57
Section 5: Interpersonal Messaging Packages		61
5.1	Class Hierarchy	61
5.2	Class Definitions	62
5.2.1	Bilaterally Defined Body Part	62
5.2.2	Body Part	63
5.2.3	Externally Defined Body Part	63
5.2.4	G3 Fax Body Part	64
5.2.5	G4 Class 1 Body Part	64
5.2.6	General Text Body Part	65
5.2.7	IA5 Text Body Part	65
5.2.8	Interpersonal Message	66
5.2.9	Interpersonal Notification	69
5.2.10	IPM Identifier	70
5.2.11	IPMS Extension	71
5.2.12	Message Body Part	71
5.2.13	Mixed-mode Body Part	72
5.2.14	Nationally Defined Body Part	73
5.2.15	Non-receipt Notification	73
5.2.16	Office Document Architecture Body Part	74
5.2.17	OR Descriptor	74
5.2.18	Receipt Notification	75
5.2.19	Recipient Specifier	76
5.2.20	Teletex Body Part	76
5.2.21	Unidentified Body Part	77
5.2.22	Videotex Body Part	77
5.3	Syntax Definitions	78
5.3.1	Acknowledgement Mode	78
5.3.2	Auto-submitted	78
5.3.3	Discard Reason	79
5.3.4	IA5 Repertoire	79
5.3.5	Importance	79

5.3.6	Non-receipt Reason	79
5.3.7	Notification Request	80
5.3.8	Sensitivity	80
5.3.9	Videotex Syntax	80
Section 6: Message Handling Packages		81
6.1	Class Hierarchy	81
6.2	Class Definitions	83
6.2.1	Algorithm	83
6.2.2	Algorithm and Result	83
6.2.3	Asymmetric Token	84
6.2.4	Bilateral Information	84
6.2.5	Communique	85
6.2.6	Content	89
6.2.7	Delivered Message	89
6.2.8	Delivered Per-recipient DR	90
6.2.9	Delivered Per-recipient NDR	91
6.2.10	Delivered Per-recipient Report	91
6.2.11	Delivered Report	92
6.2.12	Delivery Confirmation	93
6.2.13	Delivery Envelope	93
6.2.14	Delivery Report	98
6.2.15	EITs	101
6.2.16	Expansion Record	102
6.2.17	Extensible Object	103
6.2.18	Extension	104
6.2.19	External Trace Entry	105
6.2.20	G3 Fax NBPs	107
6.2.21	General Content	108
6.2.22	Internal Trace Entry	108
6.2.23	Local Delivery Confirmation	110
6.2.24	Local Delivery Confirmations	111
6.2.25	Local NDR	111
6.2.26	Local Per-recipient NDR	111
6.2.27	Message	112
6.2.28	Message RD	114
6.2.29	MT Public Data	114
6.2.30	MTS Identifier	115
6.2.31	OR Address	116
6.2.32	OR Name	126
6.2.33	Per-recipient DR	126
6.2.34	Per-recipient NDR	127
6.2.35	Per-recipient Report	128
6.2.36	Probe	129
6.2.37	Probe RD	129
6.2.38	RD	130
6.2.39	Redirection Record	130
6.2.40	Report	131
6.2.41	Security Label	133

6.2.42	Session	134
6.2.43	Submission Results	135
6.2.44	Submitted Communique	136
6.2.45	Submitted Message	138
6.2.46	Submitted Message RD	139
6.2.47	Submitted Probe	141
6.2.48	Submitted Probe RD	141
6.2.49	Teletex NBPs	143
6.2.50	Token	143
6.2.51	Token Public Data	143
6.3	Syntax Definitions	144
6.3.1	Action	144
6.3.2	Builtin EIT	144
6.3.3	Delivery Mode	145
6.3.4	Delivery Point	146
6.3.5	Diagnostic	146
6.3.6	Explicit Conversion	149
6.3.7	Postal Mode	150
6.3.8	Postal Report	150
6.3.9	Priority	151
6.3.10	Reason	151
6.3.11	Redirection Reason	151
6.3.12	Registration	152
6.3.13	Report Request	152
6.3.14	Security Classification	153
6.3.15	Terminal Type	153
Section 7:	Secure Messaging Package	155
7.1	Class Hierarchy	155
7.2	Class Definitions	156
7.2.1	Integrity Check Basis	156
7.2.2	Origin Check Basis	156
7.2.3	Per-recipient Check Basis	157
7.2.4	Per-recipient Delivery Check Basis	158
7.2.5	Per-recipient Non-delivery Check Basis	159
7.2.6	Proof of Delivery Basis	159
7.2.7	Proof of Submission Basis	160
7.2.8	MT Secret Data	161
7.3	Syntax Definitions	162
Section 8:	Definition of Constants	163
8.1	Object Identifier for This International Standard	163
8.2	OIDs for Features and Packages	163
8.3	Interpersonal Messaging Package Value Definitions	164
8.4	Message Handling Package Value Definitions	165
8.5	Secure Messaging Package Value Definitions	169
Annex A (normative)	Functional Architecture of an API	171
A.1	Message Access Interface	171
A.2	Message Transfer Interface	175

Annex B (informative) Gateway Considerations	179
B.1 Address Translation	179
B.2 Feature Translation	180
B.3 Feature Relocation	180
B.4 Representational Translation	180
Annex C (informative) Supplementary Classes	181
C.1 Introduction	181
C.2 ISO 6937 Text Body Part	182
C.3 ISO 6937 Repertoire	182
C.4 ISO 6937 Text Body Part Protocol Definition	183
C.5 USA Nationally Defined Body Part	183
Annex D (informative) Bibliography	185
Alphabetic Topical Index	187

FIGURES

Figure A-1 – Conceptual Model of the X.400 Application API	172
Figure A-2 – Conceptual Model of the X.400 Gateway API	176

TABLES

Table 3-1 – Features and Their Names	31
Table 4-1 – Interface Datatypes Specific to MH	33
Table 4-2 – MA Interface Operations	35
Table 4-3 – MA Interface Operational Units	36
Table 4-4 – MT Interface Operations	48
Table 4-5 – MT Interface Operational Units	49
Table 4-6 – MA Interface Return Codes	58
Table 4-7 – MT Interface Return Codes	59
Table 5-1 – Bilaterally-Defined-Body-Part Attributes	63
Table 5-2 – Externally-Defined-Body-Part Attributes	63
Table 5-3 – G3-Fax-Body-Part Attributes	64
Table 5-4 – G4-Class-1-Body-Part Attributes	65
Table 5-5 – General-Text-Body-Part Attributes	65
Table 5-6 – IA5-Text-Body-Part Attributes	65
Table 5-7 – Interpersonal-Message Attributes	67
Table 5-8 – Interpersonal-Notification Attributes	70
Table 5-9 – IPM-Identifier Attributes	71
Table 5-10 – IPMS-Extension Attributes	71
Table 5-11 – Message-Body-Part Attributes	72
Table 5-12 – Mixed-mode-Body-Part Attributes	72
Table 5-13 – Nationally-Defined-Body-Part Attributes	73
Table 5-14 – Non-receipt-Notification Attributes	73

Table 5-15	– Office-Document-Architecture-Body-Part Attributes . . .	74
Table 5-16	– OR-Descriptor Attributes	75
Table 5-17	– Receipt-Notification Attributes	75
Table 5-18	– Recipient-Specifier Attributes	76
Table 5-19	– Teletex-Body-Part Attributes	77
Table 5-20	– Unidentified-Body-Part Attributes	77
Table 5-21	– Videotex-Body-Part Attributes	78
Table 6-1	– Algorithm Attributes	83
Table 6-2	– Algorithm-and-Result Attributes	83
Table 6-3	– Asymmetric-Token Attributes	84
Table 6-4	– Bilateral-Information Attributes	85
Table 6-5	– Communique Attributes	86
Table 6-6	– Delivered-Message Attributes	90
Table 6-7	– Delivered-Per-recipient-DR Attributes	90
Table 6-8	– Delivered-Per-recipient-NDR Attributes	91
Table 6-9	– Delivered-Per-recipient-Report Attributes	92
Table 6-10	– Delivered-Report Attributes	93
Table 6-11	– Delivery-Confirmation Attributes	93
Table 6-12	– Delivery-Envelope Attributes	94
Table 6-13	– Delivery-Report Attributes	99
Table 6-14	– EITs Attributes	101
Table 6-15	– External-EITs Attributes	101
Table 6-16	– Expansion-Record Attributes	102
Table 6-17	– Extensible-Object Attributes	103
Table 6-18	– Extension Attributes	105
Table 6-19	– External-Trace-Entry Attributes	105
Table 6-20	– G3-Fax-NBPs Attributes	107
Table 6-21	– General-Content Attributes	108
Table 6-22	– Internal-Trace-Entry Attributes	109
Table 6-23	– Local-Delivery-Confirmation Attributes	110
Table 6-24	– Local-Delivery-Confirmations Attributes	111
Table 6-25	– Local-NDR Attributes	111
Table 6-26	– Local-Per-recipient-NDR Attributes	112
Table 6-27	– Message Attributes	113
Table 6-28	– Message-RD Attributes	114
Table 6-29	– MT-Public-Data Attributes	115
Table 6-30	– MTS-Identifier Attributes	116
Table 6-31	– OR-Address Attributes	117
Table 6-32	– Forms of O/R Address	124
Table 6-33	– OR-Name Attributes	126
Table 6-34	– Per-recipient-DR Attributes	127
Table 6-35	– Per-recipient-NDR Attributes	127
Table 6-36	– Per-recipient-Report Attributes	128
Table 6-37	– Probe Attributes	129
Table 6-38	– Probe-RD Attributes	130
Table 6-39	– Redirection-Record Attributes	131

Table 6-40	– Report Attributes	131
Table 6-41	– Security-Label Attributes	133
Table 6-42	– Session Attributes	134
Table 6-43	– Submission-Results Attributes	135
Table 6-44	– Submitted-Communique Attributes	136
Table 6-45	– Submitted-Message Attributes	138
Table 6-46	– Submitted-Message-RD Attributes	139
Table 6-47	– Submitted-Probe Attributes	141
Table 6-48	– Submitted-Probe-RD Attributes	142
Table 6-49	– Teletex-NBPs Attributes	143
Table 6-50	– Values of the Builtin-EIT Enumeration	144
Table 7-1	– Integrity-Check-Basis Attributes	156
Table 7-2	– Origin-Check-Basis Attributes	157
Table 7-3	– Per-recipient-Check-Basis Attributes	158
Table 7-4	– Per-recipient-Delivery-Check-Basis Attributes	158
Table 7-5	– Per-recipient-Non-delivery-Check-Basis Attributes	159
Table 7-6	– Proof-of-Delivery-Basis Attributes	160
Table 7-7	– Proof-of-Submission-Basis Attributes	161
Table 7-8	– MT-Secret-Data Attributes	162
Table 8-1	– Features and Packages - OID Suffix Values	163
Table 8-2	– IM Class OID Suffixes	164
Table 8-3	– IM Attribute Type Identifier Values	165
Table 8-4	– MH Class OID Suffixes	166
Table 8-5	– MH Attribute Type Identifier Values	167
Table 8-6	– SM Class OID Suffixes	169
Table 8-7	– SM Attribute Type Identifier Values	169
Table C-1	– ISO-6937-Text-Body-Part Attributes	182
Table C-2	– USA-Nationally-Defined-Body-Part Attributes	183

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

International Standard ISO/IEC 14361 was prepared by IEEE (as IEEE Std 1224.1-1993) and was adopted, under a special “fast-track procedure”, by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, in parallel with its approval by national bodies of ISO and IEC.

Annex A forms an integral part of this International Standard. Annexes B to D are for information only.

1 Introduction

(This introduction is not a normative part of ISO/IEC 14361, Information technology—MHS-based electronic messaging—Application Program Interface (API) [Language independent], but is included for information only.)

The purpose of this International Standard is to define an Application Program Interface (API) to X.400-based electronic messaging services in terms that are independent of any particular programming language. The APIs specified in this International Standard provide for access to, and interconnection of, messaging systems whose architecture is in accordance with X.400.

10 Related Standards

This International Standard is intended to provide the basis for the definition of programming language bindings to which implementations and applications can conform. Such a language binding, for the C programming language, is contained in ISO/IEC 14365 [B5].

This International Standard, and the language bindings derived from it, are intended to be used in conjunction with the OSI abstract data manipulation API, ISO/IEC 14360 [11], which provides a general-purpose API for the creation, examination, modification, and deletion of OSI information objects.

ISO/IEC 14363 [12] contains a set of requirements to be satisfied by test methods for measuring conformance to this International Standard. They are stated in terms that are independent of any particular programming language, and they apply to test methods for measuring conformance to all standards defining programming language bindings to this International Standard.

24 Overview

This International Standard defines two APIs of architectural importance in the construction of message handling systems (MHSs). It defines an X.400 API that makes the functionality of a message transfer system (MTS) accessible to a message store (MS) or user agent (UA), or the functionality of a simple MS accessible to a UA. It also defines an X.400 Gateway API that divides a message transfer agent (MTA) into two software components, a mail-system gateway and an X.400 gateway service. The interfaces are designed to be used and implemented in conjunction with the general-purpose OSI Abstract Data Manipulation API defined in ISO/IEC 14360 [11].

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

Related Standards Activities

The following areas are under active consideration at this time, or are expected to become active in the near future, concerning extensions to this International Standard. Similar efforts can be anticipated in the future:¹⁾

- (1) Directory services
- (2) FTAM API
- (3) Verification testing methods
- (4) Network interface facilities
- (5) System administration.

This International Standard is based on IEEE Std 1224.1-1993 (B8), which was prepared by the P1224 Working Group, sponsored by the Portable Applications Standards Committee of the IEEE Computer Society.

1) A *Standards Status Report* that lists all current IEEE Computer Society standards projects is available from the IEEE Computer Society, 1730 Massachusetts Avenue NW, Washington, DC 20036-1903, USA; Telephone: +1 202 371-0101; FAX: +1 202 728-9614.

Information technology—MHS-based electronic messaging—Application Program Interface (API) [Language independent]

Section 1: General

1.1 Scope

The purpose of this International Standard is to provide APIs supporting the development of applications that are users of the message transfer system, and gateways that incorporate or use X.400 mail functionality. This includes gateways between X.400 mail networks and proprietary mail systems. The interface supports access to X.400 services.

The interface is oriented towards the needs of gateways and applications that require full access to the features of the protocols.

NOTE: There are industry-standard and proprietary APIs that can be used by simple applications that only require the main features of the protocols.

This International Standard defines two application program interfaces to X.400-based electronic messaging systems. The interfaces are:

- (1) An X.400 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
- (2) An X.400 Gateway API that divides a message transfer agent (MTA) into two software components, a messaging system gateway and an X.400 gateway service.

This International Standard provides a language-independent specification.

1.2 Normative References

The following standards contain provisions which, through reference in this text, constitute provisions of this International Standard. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this International Standard are encouraged to investigate the possibility of applying the most recent editions of the standards indicated below. Members of IEC and ISO maintain registers of currently valid International Standards.

- [1] ISO 639: 1988,¹⁾ *Code for the representation of names of languages*.
- [2] ISO 3166: 1993, *Codes for the representation of names of countries*.
- [3] ISO/IEC 6937: 1994, *Information technology—Coded graphic character set for text communication—Latin alphabet*.
- [4] ISO/IEC 10021-1: 1990 (CCITT Recommendation X.400: 1988, CCITT Recommendation X.408: 1988), *Information technology—Text communication—Message-Oriented Text Interchange Systems (MOTIS)—Part 1: System and service overview*.
- [5] ISO/IEC 10021-2: 1990 (CCITT Recommendation X.402: 1988), *Information technology—Text communication—Message-Oriented Text Interchange Systems (MOTIS)—Part 2: Overall architecture*.
- [6] ISO/IEC 10021-3: 1990 (CCITT Recommendation X.407: 1988), *Information technology—Text communication—Message-Oriented Text Interchange Systems (MOTIS)—Part 3: Abstract service definition conventions*.
- [7] ISO/IEC 10021-4: 1990 (CCITT Recommendation X.411: 1988), *Information technology—Text communication—Message-Oriented Text Interchange Systems (MOTIS)—Part 4: Message transfer system: abstract service definition and procedures*.
- [8] ISO/IEC 10021-5: 1994 (ITU-T Recommendation X.413: 1995), *Information technology—Text communication—Message-Oriented Text Interchange Systems (MOTIS)—Part 5: Message store: abstract service definition*.

NOTE: ISO/IEC 10021-5: 1994 supersedes ISO/IEC 10021-5: 1990. However, when this International Standard was under development, the previous edition was valid and this International Standard is therefore based on that edition, which is listed below:

ISO/IEC 10021-5: 1990 (CCITT Recommendation X.413: 1988), *Information technology—Text communication—Message-Oriented Text Interchange Systems (MOTIS)—Part 5: Message store: abstract service definition*.
- [9] ISO/IEC 10021-6: 1990 (CCITT Recommendation X.419: 1988), *Information technology—Text communication—Message-Oriented Text Interchange Systems (MOTIS)—Part 6: Protocol specifications*.

1) ISO/IEC documents can be obtained from the ISO Central Secretariat, 1 Rue de Varembé, Case Postale 56, CH-1211, Genève 20, Switzerland/Suisse.

- {10} ISO/IEC 10021-7: 1990 (CCITT Recommendation X.420: 1988), *Information technology—Text communication—Message-Oriented Text Interchange Systems (MOTIS)—Part 7: Interpersonal messaging system.*
- {11} ISO/IEC 14360: 1996, *Information technology—Open Systems Interconnection (OSI) abstract data manipulation—Application Program Interface (API) [Language independent].*
- {12} ISO/IEC 14363: 1996, *Information technology—Test methods for measuring conformance to MHS-based electronic messaging—Application Program Interface (API) [Language independent].*
- {13} ISO/IEC 14392: 1996, *Information technology—Directory services—Application Program Interface (API) [Language independent].*
- {14} CCITT Recommendation E.163: 1988,²⁾ *Numbering plan for the international telephone services.*
- {15} CCITT Recommendation E.164: 1988, *Numbering plan for the ISDN era.*
- {16} CCITT Recommendation F.1: 1984, *Operational provisions for the international public telegram service.*
- {17} CCITT Recommendation F.170: 1988, *Operational provisions for the international public facsimile service between public bureau (bureaufax).*
- {18} CCITT Recommendation F.200: 1984, *Teletex service.*
- {19} CCITT Recommendation T.4: 1984, *Standardisation of group 3 facsimile apparatus for document transmission.*
- {20} CCITT Recommendation T.5: 1984, *General aspects of group 4 facsimile apparatus.*
- {21} CCITT Recommendation T.6: 1984, *Facsimile coding schemes and coding control functions for group 4 facsimile apparatus.*
- {22} CCITT Recommendation T.30: 1984, *Procedures for document facsimile transmission in the general switched telephone network.*
- {23} CCITT Recommendation T.50: 1984, *International Reference Alphabet.*
- {24} CCITT Recommendation T.60: 1984, *Terminal equipment for use in the teletex service.*
- {25} CCITT Recommendation T.61: 1984, *Character repertoire and coded character sets for the international teletex service.*
- {26} CCITT Recommendation T.62: 1984, *Control procedures for teletex and group 4 facsimile services.*
- {27} CCITT Recommendation T.100: 1984, *International information exchange for interactive videotex.*

²⁾ CCITT documents can be obtained from the Telecommunication Standardization Bureau of the International Telecommunication Union, Sales Section, Place des Nations, CH-1211, Genève 20, Switzerland/Suisse.

- 101 {28} CCITT Recommendation T.101: 1984, *International interworking for*
videotex services.
- 102 {29} CCITT Recommendation T.400: 1984, *Introduction to document architecture,*
 103 *transfer, and manipulation.*
- 104 {30} CCITT Recommendation T.411: 1988, *Open document architecture (ODA)*
 105 *and interchange format—introduction and general principles.*
- 106 {31} CCITT Recommendation T.501: 1984, *A document application profile MM for*
 107 *the interchange of formatted mixed-mode documents.*
- 108 {32} CCITT Recommendation T.503: 1984, *A document application profile for the*
 109 *interchange of Group 4 facsimile documents.*
- 110 {33} CCITT Recommendation X.121: 1984, *International numbering plan for pub-*
 111 *lic data networks.*
- 112 {34} CCITT Recommendation X.400: 1984, *Message handling systems: System*
 113 *model, service elements.*
- 114 {35} CCITT Recommendation X.401: 1984, *Message handling systems: Basic ser-*
 115 *vice elements and optional user facilities.*
- 116 {36} CCITT Recommendation X.408: 1984, *Message handling systems: Encoded*
 117 *information type conversion rules.*
- 118 {37} CCITT Recommendation X.409: 1984, *Message handling systems: Presenta-*
 119 *tion transfer syntax and notation.*
- 120 {38} CCITT Recommendation X.410: 1984, *Message handling systems: Remote*
 121 *operations and reliable transfer service.*
- 122 {39} CCITT Recommendation X.411: 1984, *Message handling systems: Message*
 123 *transfer layer.*
- 124 {40} CCITT Recommendation X.420: 1984, *Message handling systems: Interper-*
 125 *sonal messaging user agent layer.*
- 126 {41} CCITT Recommendation X.430: 1984, *Message handling systems: Access pro-*
 127 *tol for teletex terminals.*
- 128 {42} CEN/CENELEC ENV 41 202,³⁾ *Message Handling Systems; User Agent (UA)*
 129 *and Message Transfer Agent (MTA): Access to an Administration Manage-*
 130 *ment Domain (ADMD), August 1987.*

131 3) CEN/CENLEC publications can be obtained from CEN/CENELEC, 2 Rue Brederode Bte.5, B-
 132 1000 Brussels, Belgium.

1.3 Conformance

1.3.1 Implementation Conformance

1.3.1.1 Conformance Requirements

A conforming implementation for a programming language binding specification for this International Standard shall meet all of the following criteria:

- (1) **Support of defined behavior.** The implementation shall support all required behavior defined in this International Standard.
- (2) **Support of nonstandard extensions.** The implementation may provide additional operations or facilities not required by this International Standard or by the programming language binding specification. Each such nonstandard extension shall be identified as such in the system documentation. Nonstandard extensions, when used, may change the behavior of operations or facilities defined by this International Standard or by the programming language binding specification. The conformance document shall define an environment in which an application can be run with the behavior specified by this International Standard and the programming language binding specification. In no case shall such an environment require modification of a Strictly Conforming Application.
- (3) **API requirements.** The manufacturer and product shall satisfy the following requirements related to APIs:
 - The manufacturer shall identify the APIs (MA, MT, or both) the product implements and state what roles (client, service, or both) it plays for each. Those APIs shall support the behavior described in this International Standard and in the programming language binding specification.
 - The product shall implement the OM API defined in ISO/IEC 14360 {11},⁴⁾ satisfying its conformance requirements, and implement the same roles for that API as it implements for the MA and MT interfaces.
- (4) **Features.** The manufacturer and product shall satisfy the following requirements related to features:
 - If the product implements the role of service for the MA or MT interface, the manufacturer shall state which features it implements.
 - If the product implements the role of service for the MA interface, it shall implement the Basic Access Functional Unit (FU) and the Submission FU, the Delivery FU, the Retrieval FU, or any combination of these.
 - If the product plays the role of service for the MT interface, it shall implement the Basic Transfer FU and the Transfer In FU, the Transfer Out FU, or both.

4) The numbers in curly brackets correspond to those of the references in 1.2.

— If the product plays the role of service for either interface, the product shall implement the MH 88 Package, the MH 84 Package, or both.

(5) **Operations.** The manufacturer and product shall satisfy the following requirement related to operations:

— The product shall implement every aspect of every operation in each FU for which it plays the role of service.

(6) **Classes.** The manufacturer and product shall satisfy the following requirements related to classes:

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

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

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

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

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

— The product shall state for which classes it provides the Encode and Decode operations.

(7) **Protocols.** The manufacturer and product shall satisfy the following requirements related to protocols:

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

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

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

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

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

(8) **Options.** The manufacturer and product shall satisfy the following requirement related to implementation options:

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

(9) **Interpretation of *any* syntax.** Wherever an *any* appears in the syntax column of an attribute definition, this shall be treated as the corresponding OM syntax wherever the underlying ASN.1 encoding is a Universal simple type, as listed in ISO/IEC 14360 (11).

1.3.1.2 Conformance Documentation

A conformance document with the following information shall be available for an implementation claiming conformance to a programming language binding specification for this International Standard. The conformance document shall be in two parts. The first part shall have the same structure as this International Standard, with the information presented in the appropriately numbered sections, clauses, and subclauses. The second part shall have the same structure as the programming language binding specification, with the information presented in the appropriately numbered sections, clauses, and subclauses. The conformance document shall not contain information about extended features or capabilities outside the scope of this International Standard and the programming language binding specification.

The conformance document shall identify the programming language binding specification to which the implementation conforms.

The conformance document shall contain a statement that indicates the full names, numbers, and dates of the language-independent and programming language binding specification standards that apply.

The conformance document shall state which of the optional features defined in this International Standard and in the programming language binding specification are supported by the implementation.

The conformance document shall describe the behavior of the implementation for all implementation-defined features defined in this International Standard and in the programming language binding specification. This requirement shall be met by listing these features and by providing either a specific reference to the system documentation or full syntax and semantics of these features. The conformance document may specify the behavior of the implementation for those features where this International Standard or the programming language binding specification states that implementations may vary or where features are identified as undefined or unspecified.

No specifications other than those specified by this International Standard and the programming language binding specification shall be present in the conformance document.

The phrases “shall document” or “shall be documented” in this International Standard or in a programming language binding specification for this International Standard mean that documentation of the feature shall appear in the conformance

document, as described previously, unless the system documentation is explicitly mentioned.

The system documentation should also contain the information found in the conformance document.

1.3.1.3 Conforming Implementation Options

The following aspects of the behavior of the service are implementation defined:

- (1) The circumstances that define abnormal termination of a session
- (2) The features of a session that the `mx_open` operation supplies as defaults
- (3) Whether the service will accept any circumstances as causes of temporary delivery failure
- (4) The maximum number of users that may be assigned to a single delivery queue
- (5) The maximum number of sessions that may exist simultaneously
- (6) Which qualifying object in the delivery or input queue (if there are several such objects) the `ma_start_delivery` or `mt_start_transfer_in` operation selects for delivery or transfer in
- (7) Which and how many sessions the `mx_wait` operation notifies of the arrival of an object when several sessions provide access to a delivery, retrieval, or input queue.

1.3.2 Application Conformance

All applications claiming conformance to a programming language binding specification for this International Standard shall fall within one of the categories defined in the following subclauses.

1.3.2.1 Strictly Conforming Application

A Strictly Conforming Application is an application that requires only the facilities described in this International Standard, in the programming language binding specification, and in the applicable language standards. Such an application shall accept any behavior described in this International Standard or in the programming language binding specification as unspecified or implementation defined and, for symbolic constants, shall accept any value in the ranges permitted by this International Standard and the programming language binding specification.

1.3.2.2 Conforming Application

There is only one type of Conforming Application.

1.3.2.2.1 ISO/IEC Conforming Application

An ISO/IEC Conforming Application of a programming language binding specification for this International Standard is an application that uses only the facilities described in this International Standard, in the programming language

binding specification, in the applicable language standards, and in other language binding standards approved by ISO or IEC. Such an application shall include a statement of conformance that documents all options and limit dependencies, and all other ISO or IEC standards used.

1.3.2.3 Conforming Application Using Extensions

A Conforming Application Using Extensions of a programming language binding specification for this International Standard is an application that differs from a Conforming Application only in that it uses nonstandard facilities that are consistent with this International Standard and with the programming language binding specification. Such an application shall fully document its requirements for these extended facilities, in addition to the documentation required of a Conforming Application.

1.4 Test Methods

Any measurement of conformance to a programming language binding specification for this International Standard shall be performed using test methods that conform to ISO/IEC 14363 (12) and to any additional requirements that may be imposed by the programming language binding specification.

Section 2: Terminology and General Requirements

2.1 Conventions

2.1.1 General and Typographical Conventions

Language-independent concrete OM class names, OM attribute names, and OM attribute value names are spelled with hyphens between words (e.g., **conversion-loss-prohibited**). The first letters of language-independent OM class and OM attribute names are capitalized (e.g., **Redirection-Record**).

Language-independent datatype, operation, argument, and error names are lower-cased and are spelled with underscores between words (e.g., `ma_start_delivery`).

The use of fonts in this International Standard is as follows.

- The Helvetica font is used for:

- Language-independent operation names, such as `ma_start_delivery`

- Language-independent datatype names, such as `mh_interval_type`

- Language-independent error names, such as `feature_unavailable`

- The *italic* font is used for:

- Language-independent operation arguments, such as *user_address*

- The introduction of important terms

- Cross-references in 2.2

- The **bold** font is used for:

- Language-independent concrete OM class names, such as **Delivered-Report**

- Language-independent OM attribute names, such as **Algorithm-Datum**

- Language-independent OM attribute values, such as **conversion-loss-prohibited**

- The ***bold italic*** font is used for:

- Language-independent abstract OM class names, such as ***Delivery-Report***

— The constant width (Courier) font is used for:

— References to terms defined in standards upon which this International Standard is based.

2.1.2 Language-Independent Conventions

NOTE: In the absence of a normative notation for language-independent specifications, these conventions are based upon a draft of the POSIX language-independent specifications conventions which are being developed in IEEE PASC (B9).

This subclause defines the notation used within this International Standard to specify the information that is communicated through interfaces to the defined services. The notation assumes a model of an abstract procedural interface, although language bindings to this International Standard may take other forms.

The key elements of the notation are *datatype* definitions, *name* definitions, and *operation* definitions. The semantics of each of these elements is defined in 2.2.2.

NOTE: The notations specified are supplemented by additional descriptions using other notational schemes (including natural language). For instance, for the specification of each language-independent operation, the notation described below is used in the **Synopsis** subclause, while the semantics are described in the **Description** subclause using natural language.

In particular, in the **Descriptions** subclause, a notation is used for integer value ranges where a "(" or ")" indicates an *exclusive* range limit, whereas a "[" or "]" indicates an *inclusive* range limit. Thus the range $[0, 2^{16})$ specifies integers from 0 to $65536 - 1$ inclusive.

2.1.2.1 Notation

A modified backus-naur form (BNF) notation is used in this subclause. The characters ":", "=", "<", ">", "[", "{", and "}" are always part of this notation and are never used in terminal strings.

Nonterminal elements consist of a "<" character followed by an identifier followed by a ">" character. Each production consists of a nonterminal followed by the characters "::=" followed by a right-hand-side expression. In a right-hand-side expression, the "[" character separates alternate elements, and the "{" and "}" characters delimit optional elements.

2.1.2.2 Identifiers

```

<identifier> ::= <initial-char> <identifier-body>
<initial-char> ::= <uppercase-letter> | <lowercase-letter>
<identifier-body> ::= <body-char> { <identifier-body> }
<body-char> ::= <uppercase-letter> | <lowercase-letter> |
               <digit> | <separator>
<lowercase-letter> ::= a | b | c | d | e | f | g | h | i |
                       j | k | l | m | n | o | p | q | r | s | t |
                       u | v | w | x | y | z
<uppercase-letter> ::= A | B | C | D | E | F | G | H | I |
                     J | K | L | M | N | O | P | Q | R | S | T |

```

```

67         U | V | W | X | Y | Z
        <digit> ::= 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0
68        <separator> ::= - | _

```

69 2.1.2.3 Numbers

```

70        <number> ::= <digit> | <digit> <number>
71        <signed-number> ::= { - } <number>

```

72 2.1.2.4 Datatypes

73 The following notation is used to specify a datatype:

```

74        Datatype:      <datatype-name>
75        Definition:     <datatype>
76        Description:    ...

```

77 where

```

78        <datatype> ::= <datatype-name> | <datatype-definition>
79        <datatype-name> ::= <identifier>
80        <datatype-definition> ::= <atomic-type> | <constructed-type>
81        <atomic-type> ::= boolean_type | <opaque-type> |
82        <state-type> | <integer-type>
83        <opaque-type> ::= opaque [ <size> ]
84        <state-type> ::= state [ <state-list> ]
85        <state-list> ::= <state-value> { , <state-list> }
86        <state-value> ::= <identifier>
87        <size> ::= <number>
88        <integer-type> ::= integer_type [ <minimum> , <maximum> ]
89        <minimum> ::= <signed-number>
90        <maximum> ::= <signed-number>
91
92        <constructed-type> ::= <array-type> | <record-type> |
93        <set-type> | <choice-type> | <list-type>
94        <array-type> ::= array <index-type> of <base-type>
95        <index-type> ::= <integer-type> | <state-type>
96        <base-type> ::= <datatype>
97        <record-type> ::= record [ <member-list> ]
98        <choice-type> ::= choice [ <member-list> ]
99        <set-type> ::= set [ <datatype> ]
100        <list-type> ::= list [ <datatype> ] { max members <number> }
101        <member-list> ::= <member-definition> { , <member-list> }
102        <member-definition> ::= <member-name> : <member-type>
103        <member-name> ::= <identifier>
104        <member-type> ::= <datatype>

```

104 Any of the values within square brackets may be listed as *<implementation*
105 *defined>*; see 2.2.1.2 for the requirements imposed by this notation.

2.1.2.5 Names

The following notation is used to specify a name:

Name:	<identifier>
Datatype:	<datatype-definition>
Description:	...

2.1.2.6 Operations

The following notation is used to specify a operation:

Operation:	<identifier>
Status Datatype:	<datatype>
Input:	<parameter-name> : <datatype>
	<parameter-name> : <datatype>
	...
	...
Output:	<parameter-name> : <datatype>
	<parameter-name> : <datatype>
	...
	...

where

<parameter-name> ::= <identifier>

2.2 Definitions

2.2.1 Terminology

For the purposes of this International Standard, the following definitions apply:

2.2.1.1 conformance document: A document provided by an implementor that contains implementation details as described in 1.3.1.2. [ISO/IEC 9945-1 (B4)]

2.2.1.2 implementation defined: An indication that the implementation shall define and document the requirements for correct program constructs and correct data of a value or behavior. [ISO/IEC 9945-1 (B4)]

2.2.1.3 may: An indication of an optional feature.

With respect to implementations, the word *may* is to be interpreted as an optional feature that is not required in this International Standard, but that can be provided. With respect to Strictly Conforming Applications, the word *may* means that the optional feature shall not be used. [ISO/IEC 9945-1 (B4)]

2.2.1.4 shall: An indication of a requirement on the implementation or on Strictly Conforming Applications, where appropriate. [ISO/IEC 9945-1 (B4)]

2.2.1.5 should:

- (1) With respect to implementations, an indication of an implementation recommendation, but not a requirement.
- (2) With respect to applications, an indication of a recommended programming practice for applications and a requirement for Strictly Conforming Applications. [ISO/IEC 9945-1 {B4}]

2.2.1.6 supported: A condition regarding optional functionality.

Certain functionality in this International Standard is optional, but the interfaces to that functionality are always required. If the functionality is *supported*, the interfaces work as specified by this International Standard (except that they do not return the error condition indicated for the unsupported case). If the functionality is not *supported*, the interface shall always return the indication specified for this situation. [ISO/IEC 9945-1 {B4}]

2.2.1.7 system documentation: All documentation provided with an implementation, except the conformance document.

Electronically distributed documents for an implementation are considered part of the system documentation. [ISO/IEC 9945-1 {B4}]

2.2.1.8 undefined: An indication that this International Standard imposes no portability requirements on an application's use of an indeterminate value or its behavior with erroneous program constructs or erroneous data.

Implementations (or other standards) may specify the result of using that value or causing that behavior. An application using such behaviors is using extensions, as defined in 1.3.2.3. [ISO/IEC 9945-1 {B4}]

2.2.1.9 unspecified: An indication that this International Standard imposes no portability requirements on applications for correct program constructs or correct data regarding a value or behavior.

Implementations (or other standards) may specify the result of using that value or causing that behavior. An application requiring a specific behavior, rather than tolerating any behavior when using that functionality, is using extensions, as defined in 1.3.2.3. [ISO/IEC 9945-1 {B4}]

2.2.2 General Terms

For the purposes of this International Standard, the following definitions apply:

2.2.2.1 1988 class: A class for which all the specific attributes are for the 1988 version of the CCITT X.400 Recommendation alone.

2.2.2.2 abstract class: A class, instances of which are forbidden unless they belong to one of its concrete subclasses. [ISO/IEC 14360 {11}]

2.2.2.3 array: A datatype constructed from a base datatype and an index datatype. The base datatype may be any datatype; the index datatype shall be a finite, ordered datatype. [TCOS-LIS/D4 {B9}].

2.2.2.4 assertion: a statement that is derived from the standard to which conformance is being measured, that is true for a conforming implementation, and that pertains either to functionality or behavior of a functional interface or namespace allocation or to the documentation associated with the implementation being tested. [ISO/IEC 14360 {11}]

2.2.2.5 attempted domain: The MD to which a tracing domain attempts but fails to transfer a communicate or report.

2.2.2.6 attempted MTA: The MTA to which a tracing MTA attempts but fails to transfer a communicate or report.

2.2.2.7 attribute: A component of an object, comprised of an integer denoting the type of the attribute and an ordered sequence of one or more attribute values, each accompanied by an integer denoting the syntax of the value. [ISO/IEC 14360 {11}]

2.2.2.8 Boolean: A datatype with two values, named "TRUE" and "FALSE", and a set of logical operations: NOT, OR, AND, etc. [TCOS-LIS/D4 {B9}].

2.2.2.9 choice: A datatype constructed from a sequence of base datatypes, each associated with a name. A choice value contains, for exactly one name, a value of the corresponding datatype. [TCOS-LIS/D4 {B9}].

2.2.2.10 class: A category into which objects are placed on the basis of both their purpose and their internal structure. [ISO/IEC 14360 {11}]

2.2.2.11 client instance: A manifestation of the client that shares the input and output queues of the client with other instances.

2.2.2.12 client: Software that uses an interface. [ISO/IEC 14360 {11}]

2.2.2.13 concrete: A class, instances of which are permitted either by direct instantiation or the instantiation of its concrete subclasses. [ISO/IEC 14360 {11}]

2.2.2.14 datatype: A collection of distinguished values, together with a collection of characterizing operations on those values. [ISO/IEC 14360 {11}]

2.2.2.15 delivery queue: One of two alternative databases that the service uses to convey objects to the client of the MA interface.

2.2.2.16 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.

2.2.2.17 feature: A negotiable aspect of an interface.

207 **2.2.2.18 functional unit:** A group of related functions.

208 **2.2.2.19 generic interface:** A version of an interface that is independent of any
209 particular programming language.

210 **2.2.2.20 input queue:** The database that the service uses to convey objects to the
211 client of the MT interface.

212 **2.2.2.21 interface:** Either the MA interface or the MT interface without distinc-
213 tion, or one of the two in particular.

214 **2.2.2.22 list:** A datatype constructed from a base datatype. A list value contains a
215 sequence of zero or more values of the base datatype.

216 Applying an operation to all members of a list may be supported through either of
217 two programming paradigms. In the first, the sequencing control is provided by
218 the application; in the second, it is provided by the implementation. [TCOS-LIS/D4
219 {B9}].

220 **2.2.2.23 local environment:** The environment of the local MTA.

221 **2.2.2.24 local MD:** The MD of which the local MTA is a part.

222 **2.2.2.25 local MTA:** The MTA that comprises the client and the service in the
223 context of the MT interface.

224 **2.2.2.26 MA interface:** The X.400 Application API.

225 **2.2.2.27 mail system gateway:** Software that uses the MT interface (the client).

226 **2.2.2.28 MOTIS:** The set of ISO standards for Message-Oriented Text Inter-
227 change Systems. Defined in references {4} – {10}.

228 **2.2.2.29 MT interface:** The X.400 Gateway API.

229 **2.2.2.30 name:** An identifier used to represent a specific value of a specific
230 datatype.

231 This International Standard does not specify when a name is bound to its value: at
232 compile time, link time, or load time. The use of a name as a parameter in a data-
233 type specification does not imply that its value is defined at compile time.
234 [ISO/IEC 14360 {11}]

235 **2.2.2.31 neighboring MTA:** An MTA with which the service is prepared to estab-
236 lish OSI connections in the context of the MT interface.

2.2.2.32 network address attributes: The **ISDN-Number**, **ISDN-Subaddress**, **Presentation-Address**, and **X.121-Address** attributes specific to the **OR-Address** class.

2.2.2.33 object: Any of the complex information objects created, examined, modified, or destroyed by means of the OM interface. [ISO/IEC 14360 {11}]

2.2.2.34 OM interface: The API to OSI Object Management. [ISO/IEC 14360 {11}]

2.2.2.35 opaque: A datatype with no order or other operations defined.

An opaque datatype may have associated names that identify distinguished values. [TCOS-LIS/D4 {B9}].

2.2.2.36 operation: A program abstraction with formal input and output parameters that are bound to objects or values in the application that invokes it. [ISO/IEC 14360 {11}]

2.2.2.37 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.

2.2.2.38 originating domain: The MD at which a communique or report originated.

2.2.2.39 output queue: The database that the client of the MT interface uses to convey objects to the service.

2.2.2.40 package: A group of related classes. [ISO/IEC 14360 {11}]

2.2.2.41 personal name attributes: The **Given-Name**, **Initials**, **Surname**, and **Generation** attributes specific to the **OR-Address** class.

2.2.2.42 programming language binding specification: For a language-independent specification, a document that specifies, in terms of a particular programming language, the behavior that the language-independent specification specifies in language-independent terms.

It may also specify additional behavior that is relevant to the usage of the particular programming language. [ISO/IEC 14360 {11}]

2.2.2.43 record: A datatype constructed from a sequence of base datatypes, each associated with a name. A record value contains, for each name, a value of the corresponding base datatype. [TCOS-LIS/D4 {B9}].

268 **2.2.2.44 reserved:** An object in the delivery, retrieval, or input queue that the
269 client can access without first removing it from that queue, but that no other client
can access simultaneously.

270 **2.2.2.45 retrieval queue:** One of two alternative databases that the service uses
271 to convey objects to the client of the MA interface.

272 **2.2.2.46 service:** Software that implements an interface.

273 **2.2.2.47 service interface:** The interface as realized, for the benefit of the client,
274 by the service as a whole. [ISO/IEC 14360 {11}]

275 **2.2.2.48 set:** A datatype constructed from a base datatype; a value of a set data-
276 type contains an unordered collection of values of the base datatype, each occurring
277 only once.

278 The is-member operation returns a Boolean value that depends on whether the
279 specified value is a member of the set. Applying an operation to all members of a
280 set may be supported through either of two programming paradigms. In the first,
281 the sequencing control is provided by the application; in the second, it is provided
282 by the implementation. [TCOS-LIS/D4 {B9}].

283 **2.2.2.49 source domain:** The MD that supplies a piece of trace information.

284 **2.2.2.50 State:** An unordered, finite datatype. Each state value is identified by an
285 associated name. [TCOS-LIS/D4 {B9}].

286 **2.2.2.51 status datatype:** An abstract datatype whose values may be bound to
287 “control” values as well as “data” values. [ISO/IEC 14360 {11}]

288 **2.2.2.52 structured:** A postal O/R address that specifies the postal address of a
289 user by means of several attributes. Its structure is prescribed in some detail.

290 **2.2.2.53 subject domain:** The MD that contains the MTA embodied by the client
291 and service of the MT interface.

292 **2.2.2.54 subject message:** When used in reference to a communicate, the com-
293 municate, if it is a message; or any of the messages denoted by the communicate, if
294 it is a probe.

295 **2.2.2.55 submission queue:** The database that the client of the MA interface
296 uses to convey objects to the service.

297 **2.2.2.56 test methods:** The software, procedures, or other means specified to
298 measure conformance to a specification. [ISO/IEC 14360 {11}]

299 **2.2.2.57 tracing domain:** The MD that produces an external trace entry.

2.2.2.58 tracing MTA: The MTA that produces an internal trace entry.

300 **2.2.2.59 unstructured:** A postal O/R address that specifies the postal address of
301 a user in a single attribute. Its structure is left largely unspecified.

302 **2.2.2.60 X.400:** The set of CCITT Recommendations on message handling sys-
303 tems. This term covers both the X.400 (1984) and X.400 (1988) recommendations.

304 **2.2.2.61 X.400 (1984):** The set of CCITT Recommendations on message handling
305 systems approved in 1984. Defined in references {34} – {41}.

306 **2.2.2.62 X.400 (1988):** The set of CCITT Recommendations on message handling
307 systems approved in 1988. Defined in references {4} – {10}.

308 **2.2.2.63 X.400 Application API:** The interface that makes the functionality of
309 the MTS accessible to an MS or a UA, or the functionality of a simple MS accessi-
310 ble to a UA.

311 **2.2.2.64 X.400 Gateway API:** The interface that divides an MTA into two
312 software components, a mail system gateway and an X.400 gateway service.

313 **2.2.2.65 X.400 gateway service:** Software that implements the MT interface
314 (the service).

315 **2.2.3 Abbreviations**

316 For the purposes of this International Standard, the following abbreviations apply:

317 **2.2.3.1 1984:** X.400 (1984).

318 **2.2.3.2 1988:** X.400 (1988).

319 **2.2.3.3 ADMD:** Administration Management Domain.

320 **2.2.3.4 API:** Application Program Interface.

321 **2.2.3.5 ASN.1:** Abstract Syntax Notation One.

322 **2.2.3.6 AU:** Access Unit.

323 **2.2.3.7 BER:** Basic Encoding Rules.

324 **2.2.3.8 C:** Conditional.

325 **2.2.3.9 CA:** Certification Authority.

- 326 **2.2.3.10 CCITT:** International Telegraph and Telephone Consultative Committee.
- 327 **2.2.3.11 CDS:** Command Document Start.
- 328 **2.2.3.12 DCS:** Digital Command Signal.
- 329 **2.2.3.13 DDA:** Domain-Defined Attribute.
- 330 **2.2.3.14 DL:** Distribution List.
- 331 **2.2.3.15 DR:** Delivery Report.
- 332 **2.2.3.16 DS:** Directory System.
- 333 **2.2.3.17 EDI:** Electronic Data Interchange.
- 334 **2.2.3.18 EIT:** Encoded Information Type.
- 335 **2.2.3.19 FIF:** Facsimile Information Field.
- 336 **2.2.3.20 FU:** Functional Unit.
- 337 **2.2.3.21 G3:** Group 3 (facsimile).
- 338 **2.2.3.22 G4:** Group 4 (facsimile).
- 339 **2.2.3.23 IA5:** International Alphabet No. 5.
- 340 **2.2.3.24 IM:** Interpersonal Messaging.
- 341 **2.2.3.25 IPM:** Interpersonal Message.
- 342 **2.2.3.26 IPN:** Interpersonal Notification.
- 343 **2.2.3.27 ISDN:** Integrated Services Digital Network.
- 344 **2.2.3.28 ITA2:** International Telex Alphabet No. 2.
- 345 **2.2.3.29 M:** Mandatory.
- 346 **2.2.3.30 MA:** Message Access.
- 347 **2.2.3.31 MD:** Management Domain.

2.2.3.32 MH: Message Handling.

348 **2.2.3.33 MHS:** Message Handling System.

349 **2.2.3.34 MOTIS:** Message-Oriented Text Interchange Systems.

350 **2.2.3.35 MS:** Message Store.

351 **2.2.3.36 MT:** Message Transfer.

352 **2.2.3.37 MTA:** Message Transfer Agent.

353 **2.2.3.38 MTS:** Message Transfer System.

354 **2.2.3.39 NBP:** Nonbasic Parameter.

355 **2.2.3.40 NDR:** Nondelivery Report.

356 **2.2.3.41 NRN:** Nonreceipt Notification.

357 **2.2.3.42 O/R:** Originator/Recipient.

358 **2.2.3.43 OM:** Object Management.

359 **2.2.3.44 OSI:** Open Systems Interconnection.

360 **2.2.3.45 P1:** Message Transfer Protocol.

361 **2.2.3.46 P2:** Interpersonal Messaging Protocol.

362 **2.2.3.47 P3:** Message Submission and Delivery Protocol.

363 **2.2.3.48 P7:** MS Access Protocol.

364 **2.2.3.49 PAEK:** Public Asymmetric Encryption Key.

365 **2.2.3.50 PDAU:** Physical Delivery Access Unit.

366 **2.2.3.51 PDS:** Physical Delivery System.

367 **2.2.3.52 POSIX:** Portable Operating System Interface.

368 **2.2.3.53 PRMD:** Private Management Domain.

2.2.3.54 RD: Recipient Descriptor.

369 **2.2.3.55 RN:** Receipt Notification.

370 **2.2.3.56 SAEK:** Secret Asymmetric Encryption Key.

371 **2.2.3.57 SFD:** Simple Formattable Document.

372 **2.2.3.58 SM:** Secure Messaging.

373 **2.2.3.59 UA:** User Agent.

Section 3: Overview

This section provides an overview of the interfaces and their specifications. It also overviews the facilities provided by the OSI Object Management API, and introduces the concept of features.

3.1 Purpose

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

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

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

Henceforth, the following terms are used consistently as indicated. The terms *MA interface* and *MT interface* denote the X.400 Application API and the X.400 Gateway API, respectively. The term *interface* denotes either interface without distinction, or one interface in particular. If the latter, which interface is meant will be clear from the context. The term *service* denotes software that implements an interface. A service that implements the MA interface realizes the MTS. A service that implements the MT interface realizes an X.400 gateway service. The term *client* denotes software that uses an interface. A client that uses the MA interface

realizes a UA or an MS. A client that uses the MT interface realizes a mail system gateway.

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

3.2 Overview

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

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

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

3.3 Object Management

The interfaces are designed to be used and implemented in conjunction with the use and implementation of the general-purpose Object Management (OM) API defined in ISO/IEC 14360 (11). Throughout this International Standard, the term *OM interface* denotes the OM API, the facilities of which are introduced briefly below.

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

3.3.1 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 OM specification, and nowhere else, and they are themselves represented by integers.

3.3.2 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 (e.g., True), or as complicated as an entire OM object (e.g., a Message).

3.3.3 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 3.3.6).

An *OM attribute* is an OM attribute type, together with an ordered sequence of one or more values. OM attributes can occur only as parts of an OM object, and the OM attribute type and values are constrained by the OM class specification of that OM object (see 3.3.5). The OM attribute type can be thought of as the name of the OM attribute.

NOTE: There is no general representation for an OM attribute; however, individual language bindings may provide datatypes to allow an application to handle attributes in a convenient manner. (for example, the *OM_descriptor* data structure in the C language binding defined by ISO/IEC 14360 (11)).

3.3.4 OM Object

An *OM object* is a collection of OM attributes, the values of which can be accessed by means of operations. The particular OM attribute types that may occur in an OM object are determined by the OM class of the OM object (see 3.3.5), as are the constraints on those OM attributes. The OM class of an OM object is determined when the OM object is created and cannot be changed.

OM objects are represented in the interface by a handle or opaque pointer.

NOTE: The internal representation of an OM object is not specified; however, individual language bindings may provide datatypes to allow an application to handle objects in a convenient manner. (for example, the *OM_descriptor_list* data structure in the C language binding defined by ISO/IEC 14360 (11)).

3.3.5 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 that is discussed below.

Each OM class of OM object has a fixed list of OM attribute types, and every OM object that 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 that may appear in instances of an OM class has two parts. Each OM class *inherits* all the OM attribute types that are permitted in its immediate superclass as legal OM attribute types. There is also a list of additional OM attribute types that are permitted in the OM class. Any subclasses of this OM class will inherit all of these OM attribute types from both lists.

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

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

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

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

NOTE: The constraint is expressed in terms of bits, octets, or characters according to the kind of string. The length of a string is indicated by the number of octets by which it is represented. The number of octets in a string with multibyte characters (e.g., T.61 Teletext) may thus exceed the value length constraint. (In a language binding, 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.

3.3.6 Package

A *package* is a set of OM classes that 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.

3.3.7 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 Section 6 defines an OM class called **OR-Name**. This OM class has an OM attribute whose value is an OM object of OM class **Name**, which is defined in the Directory Service package in the directory service API, ISO/IEC 14392 {13}. 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 that 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 ISO/IEC 14360 {11}).

3.3.8 Workspace

Details of the representation of OM objects and of the implementation of the operations 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 in the previous subclause. This OM class is used in both the Message Transfer Service and in the Directory Service API, ISO/IEC 14392 {13}. 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 OM operations that support all the OM classes of OM objects in the package-closures.

The notion of a workspace also includes the storage used to represent OM objects and management of that storage. The interested reader can refer to ISO/IEC 14360 {11} for more details on how workspaces are implemented.

The application shall obtain a workspace that supports an OM class before it creates any OM objects of that OM class. The workspaces are returned by operations in the appropriate service. For example, `mt_open` returns a workspace that supports the Message Handling package, while another operation 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. (The operation `mx_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 operation `ds_initialize` and the Message Transfer Service operation `mt_open` are likely to be the same one. The application need take no account of this, but may experience improved performance.

3.3.9 Use of Objects

OM objects are used to represent the data collections used in the interface, such as parts of a message or messaging operation results. Refer to Sections 5, 6, and 7 for the definition of these OM classes.

An important feature of the interface is that an instance of a subclass can be used wherever a particular OM class is needed. This means both that the application can supply a subclass and that the service can return a subclass. For example, the application can submit messages in any format that is defined as a subclass of the class **Submitted-Communique**, and the service returns all results of the `ma_submit` operation 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` operation 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 it as a vendor extension or will ignore all OM attribute types that 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 operations. 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.

Note that public objects returned by `om_get` are read-only and shall not be modified in any way by the application.

3.4 Features

The interfaces enable the client and service to negotiate the use of the various defined features of the MA and MT interfaces. The *features* that this International Standard defines are the functional units (see below) and packages defined by them, and aspects of those functional units and packages identified as features. A *functional unit* (FU) is a group of related operations. Other standards or other ver-

sions of this International Standard may define (e.g., proprietarily) additional features, which may (but need not) themselves be FUs or packages.

The features defined by this International Standard are identified in Table 3-1. The first column of the table identifies the feature. The second column gives the symbolic name of an object identifier that denotes the feature. The ASN.1 object identifiers that denote each feature are given in Section 8.

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 mx_-open operation of the MA or MT interface. The service provides the Basic Access FU and Basic Transfer FU respectively as default for the two interfaces.

The following takes place if the client attempts to use a feature that the service has not made available. If the client invokes an operation in the Basic Access or Basic Transfer FU, but that FU is unavailable, the behavior of the service is undefined. If the Basic Access FU is available and the client invokes a function in the Submission, Delivery, or Retrieval FU, but that FU is unavailable, the feature-unavailable error arises. If the Basic Transfer FU is available and the client invokes an operation in the Transfer In or Transfer Out FU, but that FU is unavailable, the feature-unavailable error arises. If the client supplies as an operation argument an object, one of whose subobjects is an instance of a class that is not in an available package, the no-such-class error arises.

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

Table 3-1 – Features and Their Names

Feature	Name
Basic Access FU	basic-access
Submission FU	submission
Delivery FU	delivery
Retrieval FU	retrieval
Basic Transfer FU	basic-transfer
Transfer In FU	transfer-in
Transfer Out FU	transfer-out
IM 84 Package	im-84
IM 88 Package	im-88
MH 84 Package	mh-84
MH 88 Package	mh-88
SM 88 Package	sm-88
Multiple Delivery ⁽¹⁾	multiple-delivery
General Content ⁽²⁾	general-content

NOTES:

- (1) The delivery of a message or report to several local recipients as the result of a single invocation of the ma_start_delivery operation. In the absence of this feature, the service confines itself to one recipient for each invocation.
- (2) In certain circumstances a client may require access to the value of the message content octet string, for example, when engaged in secure messaging. If this feature is negotiated then the service will return a *General Content* attribute.

Section 4: Message Handling Interfaces

This section defines the MA and MT interfaces. It specifies the operations that the service makes available to the client, the datatypes of which the arguments and results of the operations are data values, and the return codes that denote the outcomes (in particular, the exceptions) that the operations may report.

4.1 Datatypes

This section defines, and Table 4-1 lists, the datatypes of the MA and MT interfaces that are specific to MH. The interfaces also include the Boolean, Object, Object Identifier, Private Object, Return Code, String, and intermediate datatypes of the OM interface.

Table 4-1 – Interface Datatypes Specific to MH

Data Type	Description
Boolean List	The result of feature negotiation for a session.
Feature List	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.

4.1.1 Boolean List

4.1.1.1 Synopsis

Datatype: mh_boolean_list_type
Definition: list[om_boolean_type]
Description: A list of boolean values, used to indicate activated features.

4.1.2 Feature List

4.1.2.1 Synopsis

Datatype: mh_feature_list_type
Definition: list[om_object_id_type]
Description: A list of features.

4.1.3 Interval

4.1.3.1 Synopsis

Datatype: mh_interval_type
Definition: om_unsigned_integer_32_type
Description: A data value of this datatype is the integer in the interval $[0, 2^{32})$ that denotes an interval of time measured in milliseconds.

4.1.4 Object Count

4.1.4.1 Synopsis

Datatype: mh_object_count_type
Definition: om_unsigned_integer_32_type
Description: A data value of this datatype is the integer in the interval $[0, 2^{32})$ that denotes a number of objects.

4.1.5 Sequence Number

4.1.5.1 Synopsis

Datatype: mh_sequence_number_type
Definition: om_unsigned_integer_32_type
Description: A data value of this datatype is the integer in the interval $[0, 2^{31})$ (*sic*) that denotes a message or report in a retrieval queue.
 Sequence numbers are assigned in ascending order, but not necessarily consecutively. The sequence number of an object never changes, and no sequence number denotes two different objects, even at different times.

4.2 Access Operations

This section defines, and Table 4-2 lists, the operations of the MA interface.

Table 4-2 – MA Interface Operations

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

As indicated in Table 4-2, the MA interface is comprised of a number of operations whose purpose and range of capabilities are summarized as follows:

Cancel Submission	This operation cancels a submitted message, with the deferred delivery option.
Close	This operation terminates an MA session between the client and the service. If the delivery or retrieval of a message or a report is in progress, the service first unsuccessfully finishes that delivery or retrieval.
Finish Delivery	This operation concludes the delivery in progress in a session. The client supplies delivery confirmations, as required, for users to which the object, a message, was delivered. It also indicates to which users the object, either a message or a report, is undeliverable.
Finish Retrieval	This operation concludes the retrieval in progress in a session. The client indicates whether the service is to remove the retrieved message or report from the retrieval queue or leave it there.
Open	This operation 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 name of the client is specified, or to a group of local users statically associated with the client name, if the O/R address of a local user is specified.
Size	This operation determines the number of messages and reports in the delivery or retrieval queue to which a session provides access.

Start Delivery This operation begins the delivery of a message or a report to one or more of the users associated with a session. If no messages or reports await delivery, the operation reports an exception.

Start Retrieval This operation begins the retrieval of a message or a report from the retrieval queue to which a session provides access. If no messages or reports await retrieval, the operation reports an exception.

Submit This operation submits a communicate (see 6.2.5) by adding it to the submission queue to which a session provides access. The operation first verifies the integrity of the communicate.

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

The operations are grouped into four FUs (one basic and one each for submission, delivery, and retrieval) as indicated in Table 4-3 below. (As stated previously, the Delivery and Retrieval FUs are mutually exclusive.)

Table 4-3 – MA Interface Operational Units

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

NOTE: The OM API is defined in ISO/IEC 14360 {11}.

The intent of the interface definition is that each operation is atomic, i.e., 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.

NOTE: Making such a guarantee might be prohibitively expensive.

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

4.2.1 Cancel Submission

4.2.1.1 Synopsis

Operation: `ma_cancel_submission`
Status Datatype: `om_return_code_type`
Input: `session : om_private_object_type`
`mts_identifier : om_object_type`

4.2.1.2 Description

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

4.2.1.3 Arguments In

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

mts_identifier The MTS identifier assigned to the message whose delivery is to be canceled; an instance of the **MTS-Identifier** class.

4.2.1.4 Results

status Whether the operation succeeded and, if not, why. It may be success or one of the values listed in 4.2.1.5.

4.2.1.5 Errors

`feature_unavailable`, `operation_interrupted`, `memory_insufficient`, `network_error`, `no_such_message`, `no_such_representation`, `no_such_session`, `no_such_syntax`, `no_such_type`, `not_private`, `permanent_error`, `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`.

4.2.2 Close

4.2.2.1 Synopsis

Operation: `ma_close`
Status Datatype: `om_return_code_type`
Input: `session : om_private_object_type`

4.2.2.2 Description

This operation terminates an MA session, as well as destroying the workspace, between the client and the service. If the delivery or retrieval of a message or a

report is in progress in the session, the service first unsuccessfully finishes that delivery or retrieval.

4.2.2.3 Arguments In

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

4.2.2.4 Results

status Whether the operation succeeded and, if not, why. It may be success or one of the values listed in 4.2.2.5.

4.2.2.5 Errors

operation_interrupted, *memory_insufficient*, *network_error*, *no_such_session*, *not_private*, *permanent_error*, *system_error*, *temporary_error*, or *wrong_class*.

4.2.3 Finish Delivery

4.2.3.1 Synopsis

Operation: *ma_finish_delivery*
Status Datatype: *om_return_code_type*
Input: *session* : *om_private_object_type*
delivery_confirmations : *om_object_type*
non_delivery_reports : *om_object_type*

4.2.3.2 Description

This operation concludes the delivery in progress in a session. The client should indicate to which users the object, either a message or a report, is undeliverable by supplying nondelivery reports. The client may optionally supply delivery confirmations for users to which the object, a message, was delivered. If the client does not explicitly confirm delivery or nondelivery, 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 nondelivery.

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

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

If the object is a report, the service returns the report to the delivery queue for the users to which it is temporarily undeliverable. For the users to which the report has been delivered, the service considers that it has transferred responsibility for the report to the client.

In all cases, the object handle shall remain valid on return from the operation and shall be deleted by the client when no longer required.

The client should not invoke the `om_delete` operation on the object prior to invoking this operation, otherwise the behavior of this operation shall be undefined.

4.2.3.3 Arguments In

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

delivery_confirmations 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.

non_delivery_reports Indicates the one or more users to which the object cannot be delivered; an instance of the **Local-NDR** class.

This argument is omitted if there are no such users.

4.2.3.4 Results

status Whether the operation succeeded and, if not, why. It may be success or one of the values listed in 4.2.3.5.

4.2.3.5 Errors

`feature_unavailable`, `operation_interrupted`, `memory_insufficient`, `network_error`, `no_such_object`, `no_such_representation`, `no_such_session`, `no_such_syntax`, `no_such_type`, `not_private`, `permanent_error`, `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`.

4.2.4 Finish Retrieval

4.2.4.1 Synopsis

Operation: `ma_finish_retrieval`
Status Datatype: `om_return_code_type`
Input: `session` : `om_private_object_type`
`remove` : `om_boolean_type`

4.2.4.2 Description

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

234 In the former case, the service considers that it has transferred responsibility for
235 the object from the service to the client. The object handle shall remain accessible
236 on return from this operation and can be deleted (using the `om_delete` operation)
237 when no longer required.

238 In the latter case, the objects shall be returned to the input queue and marked
239 unreserved. The associated object handle shall be made inaccessible on return from
240 the operation and should not be deleted by the client, but the associated com-
241 munique or report can be obtained again in a subsequent `ma_start_retrieval` opera-
242 tion invocation.

243 The client should not invoke the `om_delete` operation on the object prior to invok-
244 ing this operation, otherwise the behavior of this operation shall be undefined.

4.2.4.3 Arguments In

246	<i>session</i>	An established MA session between the client and the service; an
247		instance of the Session class. A retrieval shall be in progress.
248	<i>remove</i>	Whether the service is to remove the retrieved message or report
249		from the retrieval queue, rather than leave it there.

4.2.4.4 Results

251	<i>status</i>	Whether the operation succeeded and, if not, why. It may be
252		success or one of the values listed in 4.2.4.5.

4.2.4.5 Errors

254 `feature_unavailable`, `operation_interrupted`, `memory_insufficient`, `network_error`, `no_`-
255 `such_object`, `no_such_session`, `not_private`, `permanent_error`, `session_not_busy`,
256 `system_error`, `temporary_error`, or `wrong_class`.

4.2.5 Open

4.2.5.1 Synopsis

259	Operation:	<code>ma_open</code>
260	Status Datatype:	<code>om_return_code_type</code>
261	Input:	<code>user_address</code> : <code>om_public_object_type</code>
262		<code>client_name</code> : <code>om_string_type</code>
263		<code>feature_list</code> : <code>mh_feature_list_type</code>
264	Output:	<code>session</code> : <code>om_private_object_type</code>
265		<code>workspace</code> : <code>om_workspace_type</code>
266		<code>activated</code> : <code>om_boolean_list_type</code>

4.2.5.2 Description

This operation establishes an MA session between the client and the service, and it 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 name of the client is specified, or to a group of local users statically associated with the client name, if the O/R address of a local user is specified. The Retrieval FU can be requested only if a single user is designated, but that user may be designated in either of these two ways.

The client always designates a particular user in the same way. The choice between O/R address and client name is made by means outside the scope of this International Standard. How users are associated with a client name also is outside the scope of this International Standard. 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 operations invoked within that session. The workspace is used as an argument in the `om_create` and `om_copy` operations.

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

4.2.5.3 Arguments In

user_address An explicit identification of the local user to which the session is to provide MTS access; an instance of the **OR-Address** class.

If this argument is absent, the *client_name* argument shall be present.

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

If this argument is absent, the *user_address* argument shall be present.

feature_list An ordered sequence of features, each represented by an object identifier.

4.2.5.4 Arguments Out

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

activated If successful, an ordered sequence of Boolean values with the same number of elements as the *feature-list* 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.

4.2.5.5 Results

status Whether the operation succeeded and, if not, why. It may be success or one of the values listed in 4.2.5.6.

4.2.5.6 Errors

feature_conflicts, feature_unavailable, operation_interrupted, memory_insufficient, network_error, no_such_client, no_such_representation, no_such_syntax, no_such_type, no_such_user, permanent_error, 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.

4.2.6 Size

4.2.6.1 Synopsis

Operation: *ma_size*
Status Datatype: *om_return_code_type*
Input: *session: om_private_object_type*
Output: *number: mh_object_count_type*

4.2.6.2 Description

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

4.2.6.3 Arguments In

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

4.2.6.4 Arguments Out

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

4.2.6.5 Results

status Whether the operation succeeded and, if not, why. It may be success or one of the values listed in 4.2.6.6.