

IEEE Standard for Information Technology— X.400-Based Electronic Messaging C Language Interfaces— Binding for Application Program Interface (API)

Sponsor

**Portable Applications Standards Committee
of the
IEEE Computer Society**

Approved March 18, 1993

IEEE Standards Board

Approved September 16, 1993

American National Standards Institute

Abstract: A C language binding to IEEE Std 1224.1-1993, IEEE Standard for Information Technology— X.400-Based Electronic Messaging—Application Program Interface (API) [Language Independent], is defined. This standard is intended for use by application developers, system implementors, test method writers, and users to support application portability at the source-code level.

Keywords: API, application program interface, C language, language binding, X.400

The Institute of Electrical and Electronics Engineers, Inc.
345 East 47th Street, New York, NY 10017-2394, USA
Copyright © 1993 by the
Institute of Electrical and Electronics Engineers, Inc.
All rights reserved. Published 1993
Printed in the United States of America
ISBN 1-55937-312-1

No part of this publication may be reproduced in any form, in an electronic retrieval system or otherwise, without the prior written permission of the publisher.

IEEE Standards documents are developed within the Technical Committees of the IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Board. Members of the committees serve voluntarily and without compensation. They are not necessarily members of the Institute. The standards developed within IEEE represent a consensus of the broad expertise on the subject within the Institute as well as those activities outside of IEEE that have expressed an interest in participating in the development of the standard.

Use of an IEEE Standard is wholly voluntary. The existence of an IEEE Standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the IEEE Standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard. Every IEEE Standard is subjected to review at least every five years for revision or reaffirmation. When a document is more than five years old and has not been reaffirmed, it is reasonable to conclude that its contents, although still of some value, do not wholly reflect the present state of the art. Users are cautioned to check to determine that they have the latest edition of any IEEE Standard.

Comments for revision of IEEE Standards are welcome from any interested party, regardless of membership affiliation with IEEE. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments.

Interpretations: Occasionally questions may arise regarding the meaning of portions of standards as they relate to specific applications. When the need for interpretations is brought to the attention of the IEEE, the Institute will initiate action to prepare appropriate responses. Since IEEE Standards represent a consensus of all concerned interests, it is important to ensure that any interpretation has also received the concurrence of a balance of interests. For this reason, the IEEE and the members of its technical committees are not able to provide an instant response to interpretation requests except in those cases where the matter has previously received formal consideration.

Comments on standards and requests for interpretations should be addressed to:

Secretary, IEEE Standards Board
445 Hoes Lane
P.O. Box 1331
Piscataway, NJ 08855-1331

IEEE Standards documents are adopted by the Institute of Electrical and Electronics Engineers without regard to whether their adoption may involve patents on articles, materials, or processes. Such adoption does not assume any liability to any patent owner, nor does it assume any obligation whatever to parties adopting the standards documents.

Contents

	PAGE
Introduction	v
Section 1: General	1
1.1 Scope	1
1.2 Normative References	1
1.3 Conformance	2
1.4 Test Methods	3
Section 2: Terminology and General Requirements	5
2.1 Conventions	5
2.2 Definitions	6
2.3 General Terms	7
2.4 Abbreviations	7
Section 3: Message Handling Interfaces	9
3.1 Introduction	9
3.2 Datatypes	9
3.2.1 Feature	9
3.2.2 Interval	9
3.2.3 Object Count	10
3.2.4 Sequence Number	10
3.3 Access Functions	10
3.3.1 Cancel Submission	10
3.3.2 Close	11
3.3.3 Finish Delivery	11
3.3.4 Finish Retrieval	12
3.3.5 Open	12
3.3.6 Size	13
3.3.7 Start Delivery	13
3.3.8 Start Retrieval	14
3.3.9 Submit	14
3.3.10 Wait	15
3.4 Transfer Functions	15
3.4.1 Close	15
3.4.2 Finish Transfer In	16
3.4.3 Open	16
3.4.4 Size	17
3.4.5 Start Transfer In	18
3.4.6 Transfer Out	18
3.4.7 Wait	19
3.5 Return Codes	19

3.6 Header File	19
Section 4: Interpersonal Messaging Packages	21
Section 5: Message Handling Packages	27
Section 6: Secure Messaging Packages	39
Annex A (informative) Example <ximp.h> Header	41
Annex B (informative) Example <xmhp.h> Header	45
Annex C (informative) Example <xsmph.h> Header	55
Annex D (informative) Bibliography	57
Alphabetic Topical Index	59

TABLES

Table 2-1 – Derivation of C Identifiers	6
Table 4-1 – Identifiers for IM Package Classes	22
Table 4-2 – Identifiers for IM Package Acknowledgement Mode Enumeration	22
Table 4-3 – Identifiers for IM Package Discard Reason Enumeration	23
Table 4-4 – Identifiers for IM Package IA5 Repertoire Enumeration	23
Table 4-5 – Identifiers for IM Package Importance Enumeration	23
Table 4-6 – Identifiers for IM Package ISO 6937 Repertoire Enumeration	23
Table 4-7 – Identifiers for IM Package Non-receipt Reason Enumeration	23
Table 4-8 – Identifiers for IM Package Notification Request Enumeration	24
Table 4-9 – Identifiers for IM Package Sensitivity Enumeration	24
Table 4-10 – Identifiers for IM Package Videotex Syntax Enumeration	24
Table 4-11 – Identifiers for IM Package Attribute Types	25
Table 4-12 – Identifiers for IM Package Value Lengths	25
Table 5-1 – Identifiers for MH Package Classes	28
Table 5-2 – Identifiers for MH Package Action Enumeration	29
Table 5-3 – Identifiers for MH Package Builtin EIT Enumeration	29
Table 5-4 – Identifiers for MH Package Delivery Mode Enumeration	29
Table 5-5 – Identifiers for MH Package Delivery Point Enumeration	30
Table 5-6 – Identifiers for MH Package Diagnostic Enumeration	30
Table 5-7 – Identifiers for MH Package Explicit Conversion Enumeration	31
Table 5-8 – Identifiers for MH Package Postal Mode Enumeration	31
Table 5-9 – Identifiers for MH Package Postal Report Enumeration	31
Table 5-10 – Identifiers for MH Package Priority Enumeration	32
Table 5-11 – Identifiers for MH Package Reason Enumeration	32

Table 5-12 – Identifiers for MH Package Redirection Reason Enumeration .	32
Table 5-13 – Identifiers for MH Package Registration Enumeration . . .	32
Table 5-14 – Identifiers for MH Package Report Request Enumeration . .	32
Table 5-15 – Identifiers for MH Package Security Classification Enumeration	33
Table 5-16 – Identifiers for MH Package Terminal Type Enumeration . .	33
Table 5-17 – Identifiers for MH Package Content Type Integer	33
Table 5-18 – Identifiers for MH Package Content Type OIDs	33
Table 5-19 – Identifiers for MH Package External EITs OIDs	34
Table 5-20 – Identifiers for MH Package ODA EIT OIDs	34
Table 5-21 – Identifiers for MH Package Rendition Attributes OIDs . . .	34
Table 5-22 – Identifiers for MH Package Attribute Types	35
Table 5-23 – Identifiers for MH Package Value Lengths	37
Table 5-24 – Identifiers for MH Package Value Numbers	37
Table 6-1 – Identifiers for SM Package Classes	40
Table 6-2 – Identifiers for SM Package Attribute Types	40
Table 6-3 – Identifiers for SM Package Value Lengths	40
Table 6-4 – Identifiers for SM Package Value Numbers	40

Introduction

(This introduction is not a normative part of IEEE Std 1327.1-1993, IEEE Standard for Information Technology—X.400-Based Electronic Messaging C Language Interfaces—Binding for Application Program Interface (API), but is included for information only.)

Purpose

The purpose of this standard is to define a C programming language binding for the X.400-based electronic messaging Application Program Interface (API) that is defined in IEEE Std 1224.1-1993 {3}.

Related Standards

The programming language independent specification on which the programming language binding defined in this standard is based is contained in IEEE Std 1224.1-1993 {3}.

IEEE Std 1224.1-1993 {3}, and the language bindings derived from it, are intended to be used in conjunction with the OSI abstract data manipulation API, IEEE Std 1224-1993 {2}, which provides a general-purpose API for the creation, examination, modification, and deletion of OSI information objects.

IEEE Std 1328.1-1993 {5} specifies a set of requirements to be satisfied by test methods for measuring conformance to this standard. In addition, such test methods must also satisfy the requirements specified in IEEE Std 1326.1-1993 {4} for test methods for measuring conformance to the programming language independent specification of the API.

Overview

The API defined in this standard consists of C language bindings for the language independent datatypes and operation defined in IEEE Std 1224.1-1993 {3}. In addition, this standard provides definitions of symbolic constants and their values associated with the OM packages defined by that standard.

Acknowledgments

The base document for this standard is XO/CAE/91/100: 1991, the *API to Electronic Mail (X.400)* {B2}, copyright © 1989–1990 by X.400 API Association, United States of America, and by X/Open Company Limited, United Kingdom. The IEEE is grateful to both X/Open and the X.400 API Association for permission to use their materials.

1 **Related Standards Activities**

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

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

10 Extensions are approved as “amendments” or “revisions” to this document, follow-
11 ing IEEE Standards procedures.

12 Approved amendments are published separately and distributed from the IEEE
13 until the full document is reprinted and such amendments are incorporated in
14 their proper positions.

15 If you have any question about the completeness of your version, you may contact
16 the IEEE Standards Department [+1 (908) 562-3800] to determine what amend-
17 ments have been published.

18 If you have an interest in participating in the working groups addressing these
19 issues, please send your name, address, and phone number to the Secretary, IEEE
20 Standards Board, Institute of Electrical and Electronics Engineers, Inc., P.O. Box
21 1331, 445 Hoes Lane, Piscataway, NJ 08855-1331, USA, and ask to have this for-
22 ward to the chairperson of the appropriate PASC working group.

23
24 1) A *Standards Status Report* that lists all current IEEE Computer Society standards projects is available from
25 the IEEE Computer Society, 1730 Massachusetts Avenue NW, Washington, DC 20036-1903, USA; Telephone:
26 +1 202 371-0101; FAX: +1 202 728-9614.

1 This standard was prepared by the P1224 Working Group, sponsored by the Portable Applica-
2 tions Standards Committee of the IEEE Computer Society. At the time this standard was
3 approved, the membership of the P1224 Working Group was as follows:

4 **Portable Applications Standards Committee (PASC)**

5 Chair: Jim Isaak
6 Vice Chairs: Hal Jespersen
7 Lorraine Kevra
8 Pete Meier
9 Andrew Twigger
10 Treasurer: Peter Smith
11 Secretary: Shane McCarron

12 **P1224 Working Group Officials**

13 Chair: Steve Trus
14 Vice Chair: Tim Carter
15 Editor/Secretary: Iain Devine
16 Graham Jack

17 **P1224 Working Group**

18 Curt Campbell	Mark Karan	Pierre Perrot
19 Tony Cincotta	Eva Kuiper	Enzo Signore
20 Chris Harding	Kevin Montgomery	Daniella Sirocchi
21 Khiem Ho	Ed Owens	Andrew Twigger
22 Petr Janeček		Bruce Weiner

23 The following persons were members of the P1224 balloting group that approved
24 this standard for submission to the IEEE Standards Board:

25	Stephen R. Walli	<i>EurOpen Organizational Representative</i>	
26	Ralph Barker	<i>Uniforum Organizational Representative</i>	
27	Timothy Baker	Michel Gien	Scott E. Preece
28	Ralph Barker	Steve Griesmer	Ronald K. Sellars
29	Andy Bihain	Charles Hammons	Sandra Swearingen
30	Mack W. Bishop	Michael Hannah	James G. Tanner
31	Robert Bismuth	Mark Hazzard	Donn S. Terry
32	George Carson	Barry Hedquist	Alan E. Turner
33	Tim Carter	Khiem Ho	Mark-Rene Uchida
34	Andy Cheese	Petr Janecek	Andrew Walker
35	Fred D. Crowner	Hal Jespersen	Paul Wanish
36	Ana Maria De Alvare	Robert Knighten	Peter Wishart
37	Iain Devine	Kevin Lewis	Oren Yuen
38	Stephen L. Diamond	Roger Martin	George R. Zerdian
39	Philip H. Enslow	Joberto S.B. Martins	Jason Zions
40	Donald Folland	Diana Norwood	John Zolnowsky
41		Edward Owens	

1 The IEEE Standards Board approved this standard on March 18, 1993 with the
2 following membership:

3 **Wallace S. Read**, *Chair*

Donald C. Loughry, *Vice Chair*

4 **Andrew G. Salem**, *Secretary*

5 Gilles A. Baril

Ben C. Johnson

Don T. Michael*

6 Clyde R. Camp

Walter J. Karplus

Marco W. Migliaro

7 Donald C. Fleckenstein

Lorraine C. Kevra

L. John Rankine

8 Jay Forster*

E. G. "Al" Kiener

Arthur K. Reilly

9 David F. Franklin

Ivor N. Knight

Ronald H. Reimer

10 Ramiro Garcia

Joseph Koepfinger*

Gary S. Robinson

11 Donald N. Heirman

D. N. "Jim" Logothetis

Leonard L. Tripp

12 Jim Isaak

Donald W. Zipse

13 *Member Emeritus

14 Also included are the following nonvoting IEEE Standards Board liaisons:

15 Satish K. Aggarwal

16 James Beall

17 Richard B. Engelman

18 David E. Soffrin

19 Stanley I. Warshaw

20 Mary Lynne Nielsen

21 *IEEE Standards Project Editor*

IEEE Standard for Information Technology—X.400-Based Electronic Messaging C Language Interfaces—Binding for Application Program Interface (API)

Section 1: General

1.1 Scope

This standard defines an ISO/IEC 9989: 1990 {1} C language binding to the language-independent API defined in IEEE Std 1224.1-1993 {3}.¹⁾ It is intended to support application portability at the source-code level and to be used by application developers, system implementors, test method writers, and users.

1.2 Normative References

The following standards contain provisions that, through references in this text, constitute provisions of this standard. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this standard are encouraged to investigate the possibility of applying the most recent editions of the standards listed below.

{1} ISO/IEC 9989: 1990²⁾, *Programming languages—C*.

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

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

- 1 {2} IEEE Std 1224-1993³⁾, *IEEE Standard for Information Technology—Open*
2 *Systems Interconnection (OSI) Abstract Data Manipulation—Application*
3 *Program Interface (API) [Language Independent]*.
- 4 {3} IEEE Std 1224.1-1993, *IEEE Standard for Information Technology—*
5 *X.400-Based Electronic Messaging—Application Program Interface (API)*
6 *[Language Independent]*.
- 7 {4} IEEE Std 1326.1-1993, *IEEE Standard for Information Technology—Test*
8 *Methods for Measuring Conformance to X.400-Based Electronic*
9 *Messaging—Application Program Interface (API) [Language Independent]*.
- 10 {5} IEEE Std 1328.1-1993, *IEEE Standard for Information Technology—Test*
11 *Methods for Measuring Conformance to X.400-Based Electronic Messaging C*
12 *Language Interfaces—Binding for Application Program Interface (API)*.

13 **1.3 Conformance**

14 **1.3.1 Implementation Conformance**

15 **1.3.1.1 Requirements**

16 A conforming implementation for this standard shall be a conforming implementa-
17 tion, as defined in IEEE Std 1224.1-1993 {3}, for the programming language bind-
18 ing specification that is contained in this standard.

19 **1.3.1.2 Documentation**

20 A conformance document shall be available for an implementation claiming confor-
21 mance to this standard. It shall satisfy the requirements that are defined in IEEE
22 Std 1224.1-1993 {3} for a conformance document for a programming language
23 binding specification.

24 The phrases “shall document” or “shall be documented” in this standard mean that
25 documentation of the feature shall appear in the conformance document, unless
26 the system documentation is explicitly mentioned.

27 The system documentation should also contain the information found in the con-
28 formance document.

29 _____
30 3) IEEE documents can be obtained from the Institute of Electrical and Electronic Engineers, Service Center, 445
31 Hoes Lane, P.O. Box 1331, Piscataway, NJ, 08855-1331, USA.

1.3.1.3 Conforming Implementation Options

In addition to those requirements defined in IEEE Std 1224.1-1993 {3}, the following aspect of the behavior of the service is implementation defined:

- *Implementation of optional identifier values.* Sections 4, 5, and 6 contain C language identifiers and associated values for OM packages defined in IEEE Std 1224.1-1993 {3}. Use by an implementation of the *values* defined in these section is optional. *However*, if an implementation is to support a package and interwork with other implementations of this standard or of related standards (such as the OSI Abstract Data Manipulation API defined in IEEE Std 1224-1993 {2}) then it *shall* use the values given in the relevant section of this standard.

1.3.2 Application Conformance

All applications claiming conformance to this standard shall fall within one of the following categories.

1.3.2.1 Strictly Conforming Application

A Strictly Conforming Application for this standard is a Strictly Conforming Application as defined in IEEE Std 1224.1-1993 {3} for the programming language binding specification contained in this standard.

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 for this standard is an ISO/IEC Conforming Application as defined in IEEE Std 1224.1-1993 {3} for the programming language binding specification contained in this standard.

1.3.2.3 Conforming Application Using Extensions

A Conforming Application Using Extensions for this standard is a Conforming Application Using Extensions as defined in IEEE Std 1224.1-1993 {3} for the programming language binding specification contained in this standard.

1 **1.4 Test Methods**

2 Any measurement of conformance to this standard shall be performed
3 using test methods that conform to IEEE Std 1326.1-1993 {4} and to
4 IEEE Std 1328.1-1993 {5}.

Section 2: Terminology and General Requirements

2.1 Conventions

2.1.1 General and Typographic Conventions

When presented in **Synopsis** sections or in extended code fragments, C language source code is printed using **fixed width** font. Where C language names or code fragments appear in the text of the standard, the following conventions apply:

- A C language function, datatype, or code fragment appears in *italic* font. A function name is indicated by following parentheses, for example, *ma_submit()*.
- A symbolic constant, other than an error name, is surrounded by braces, for example, {OMP_O_MH_FE_BASIC_ACCESS}.
- A symbolic constant for an error name is surrounded by brackets, for example, [MH_RC_BAD_MESSAGE].

2.1.2 C Naming Conventions

How the identifier for an element of a C interface is derived from the name of the corresponding element of the generic interface depends upon the type of the element as specified in Table 2-1. The generic name is prefixed with the character string in the second column of the table, alphabetic characters are converted to the case in the third column, and an underscore (`_`) is substituted for each hyphen (`-`) or space ().

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

2.1.3 Language-Independent Conventions

The language-independent conventions defined in IEEE Std 1224.1-1993 {3} shall apply in this standard.

2.2 Definitions

Table 2-1 – Derivation of C Identifiers

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

NOTE: The C identifiers denote only the Elements components of data values of the OM String data type.

2.2 Definitions

2.2.1 Terminology

For the purposes of this 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: 1990 {B1}]

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: 1990 {B1}]

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 standard, but 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: 1990 {B1}]

2.2.1.4 shall: An indication of a requirement on the implementation or on Strictly Conforming Applications, where appropriate. [ISO/IEC 9945-1: 1990 {B1}]

2.2.1.5 should:

- (1) With respect to implementations, an indication of an implementation recommendation, but not a requirement.

- 1 (2) With respect to applications, an indication of a recommended program-
2 ming practice for applications and a requirement for Strictly Conforming
3 Applications. [ISO/IEC 9945-1: 1990 {B1}]

4 **2.2.1.6 supported:** A condition regarding optional functionality.

5 Certain functionality in this standard is optional, but the interfaces to that func-
6 tionality are always required. If the functionality is *supported*, the interfaces
7 work as specified by this standard (except that they do not return the error condi-
8 tion indicated for the unsupported case). If the functionality is not *supported*, the
9 interface shall always return the indication specified for this situation. [ISO/IEC
10 9945-1: 1990 {B1}]

11 **2.2.1.7 system documentation:** All documentation provided with an implemen-
12 tation, except the conformance document.

13 Electronically distributed documents for an implementation are considered part of
14 the system documentation. [ISO/IEC 9945-1: 1990 {B1}]

15 **2.2.1.8 undefined:** An indication that this standard imposes no portability
16 requirements on an application's use of an indeterminate value or its behavior
17 with erroneous program constructs or erroneous data.

18 Implementations (or other standards) may specify the result of using that value or
19 causing that behavior. An application using such behaviors is using extensions, as
20 defined in 1.3.2.3. [ISO/IEC 9945-1: 1990 {B1}]

21 **2.2.1.9 unspecified:** An indication that this standard imposes no portability
22 requirements on applications for correct program constructs or correct data
23 regarding a value or behavior.

24 Implementations (or other standards) may specify the result of using that value or
25 causing that behavior. An application requiring a specific behavior, rather than
26 tolerating any behavior when using that functionality, is using extensions, as
27 defined in 1.3.2.3. [ISO/IEC 9945-1: 1990 {B1}]

28 **2.3 General Terms**

29 The terms used in this standard are the same as those defined in IEEE Std
30 1224.1-1993 {3}.

31 **2.4 Abbreviations**

32 The abbreviations used in this standard are the same as those defined in IEEE
33 Std 1224.1-1993 {3}.

Section 3: Message Handling Interfaces

3.1 Introduction

This section defines the C language binding to the message handling interfaces in IEEE Std 1224.1-1993 {3}.

3.2 Datatypes

3.2.1 Feature

3.2.1.1 C Declaration

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

3.2.1.2 Description

This single datatype, used for negotiating the features of a session, corresponds to both the `mh_feature_list_type` and `mh_activated_list_type` datatypes defined in IEEE Std 1224.1-1993 {3}.

3.2.2 Interval

3.2.2.1 C Declaration

```
typedef OM_uint32 MH_interval;
```

3.2.2.2 Description

This datatype, used to denote an interval of time in milliseconds, corresponds to the `mh_feature_type` datatype defined in IEEE Std 1224.1-1993 {3}.

3.2.3 Object Count

3.2.3.1 C Declaration

```
typedef OM_uint32 MH_object_count;
```

3.2.3.2 Description

This datatype, an integer that denotes a number of objects, corresponds to the `mh_object_count_type` datatype defined in IEEE Std 1224.1-1993 {3}.

3.2.4 Sequence Number

3.2.4.1 C Declaration

```
typedef OM_uint32 MH_sequence_number;
```

3.2.4.2 Description

This datatype, used to denote the sequence number of an object in a retrieval queue, corresponds to the `mh_sequence_number_type` datatype defined in IEEE Std 1224.1-1993 {3}.

3.3 Access Functions

The C language functions defined in this section correspond directly to the language-independent operations of IEEE Std 1224.1-1993 {3}. They are therefore not described further in this standard, except where there is some feature of the C language binding that is not present in the language-independent specification.

3.3.1 Cancel Submission

3.3.1.1 C Declaration

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

3.3.1.2 Description

This function, used to cancel a submitted message with the deferred delivery option, corresponds to the `ma_cancel_submission` operation defined in IEEE Std 1224.1-1993 {3}.

3.3.1.3 Errors

[OM_POINTER_INVALID]

3.3.2 Close**3.3.2.1 C Declaration**

```

OM_return_code
ma_close (
    OM_private_object    session
);

```

3.3.2.2 Description

This function, used to terminate an MA session between the client and the service, corresponds to the `ma_close` operation defined in IEEE Std 1224.1-1993 {3}.

3.3.2.3 Errors

[OM_POINTER_INVALID]

3.3.3 Finish Delivery**3.3.3.1 C Declaration**

```

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

```

3.3.3.2 Description

This function, used to conclude the delivery in progress in a session, corresponds to the `ma_finish_delivery` operation defined in IEEE Std 1224.1-1993 {3}.

3.3.3.3 Arguments

delivery_confirmations

When the argument *delivery_confirmations* is omitted, its absence is signaled by the null pointer.

non_delivery_reports

When the argument *non_delivery_reports* is omitted, its absence is signaled by the null pointer.

3.3.3.4 Errors

[OM_POINTER_INVALID]

3.3.4 Finish Retrieval

3.3.4.1 C Declaration

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

3.3.4.2 Description

This function, used to conclude the retrieval in progress in a session, corresponds to the `ma_finish_retrieval` operation defined in IEEE Std 1224.1-1993 {3}.

3.3.4.3 Errors

[OM_POINTER_INVALID]

3.3.5 Open

3.3.5.1 C Declaration

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

3.3.5.2 Description

This function, used to establish an MA session between the client and the service, corresponds to the `ma_open` operation defined in IEEE Std 1224.1-1993 {3}.

3.3.5.3 Arguments

feature_list The sequence of *MH_feature* structures is terminated by an object identifier with no components (a length of zero and any value of the data pointer).