

IEEE Standard for Information Technology—Test Methods for Measuring Conformance to Open Systems Interconnection (OSI) Abstract Data Manipulation—Application Program Interface (API) [Language Independent]

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

Approved March 18, 1993
IEEE Standards Board

Abstract: Test methods for measuring conformance to IEEE Std 1224-1993, which defines a general purpose OSI Abstract Data Manipulation (OM) Application Programming Interface (API) in terms that are independent of any particular programming language, are provided.

Keywords: abstract data manipulation, API, language independent, OSI, test methods

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 1-55937-308-3

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 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 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
USA

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.

Introduction

(This introduction is not a normative part of IEEE Std 1326-1993, IEEE Standard for Information Technology—Test Methods for Measuring Conformance to Open Systems Interconnection (OSI) Abstract Data Manipulation—Application Program Interface (API) [Language Independent], but is included for information only.)

Purpose

The purpose of this standard is to define test methods for measuring conformance to IEEE Std 1224-1993 {2}, which defines a general purpose OSI Abstract Data Manipulation (OM) Application Programming Interface (API) in terms that are independent of any particular programming language.

Related Standards

IEEE Std 1224-1993 {2} is stated in terms that are independent of any particular programming language, but 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 IEEE Std 1327-1993 {B7}. The requirements specified in this standard apply to test methods for measuring conformance to standards defining all such programming language bindings of IEEE Std 1224-1993 {2}. Additional programming language-specific requirements may be defined for test methods for measuring conformance to each programming language binding of IEEE Std 1224-1993 {2}. Such a set of requirements, for test methods for measuring conformance to the C programming language binding contained in IEEE Std 1327-1993 {B7}, is contained in IEEE Std 1328-1993 {B8}.

IEEE Std 1224-1993 {2}, and the language bindings derived from it, are intended to be used in the definition of application-specific APIs that provide OSI services, such as the X.400-Based API defined in IEEE Std 1224.1-1993 {B5} and the API to Directory Services defined in IEEE Std 1224.2-1993 {B6}. The requirements specified in this standard may also apply to test methods for measuring conformance to standards defining such application-specific APIs.

The test assertions in this standard were produced in accordance with IEEE Std 1003.3-1991 {1}.

Overview

This standard contains a section corresponding to each section of IEEE Std 1224-1993 {2}. Each such section contains test assertions for that section of IEEE Std 1224-1993 {2}. Each set of test methods for measuring conformance to IEEE Std 1224-1993 {2} is required to test these assertions in conformance with the requirements of IEEE Std 1003.3-1991 {1}.

Related Standards Activities

The following areas are under active consideration at this time, or are expected to become active in the near future, concerning standards for application APIs that use the mechanism defined in this standard. Similar efforts can be anticipated in the future: ¹

- 1) X.400-based message handling
- 2) Directory services
- 3) FTAM API
- 4) Verification testing methods
- 5) Network interface facilities
- 6) System administration

¹1) 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.

Extensions are approved as “amendments” or “revisions” to this document, following IEEE Standards procedures.

Approved supplements are published separately and distributed with orders from the IEEE for this document until the full document is reprinted and such supplements are incorporated in their proper positions.

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

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

This standard was prepared by the P1224 Working Group, sponsored by the Portable Applications Standards Committee of the IEEE Computer Society. At the time this standard was approved, the membership of the P1224 Working Group was as follows:

Portable Applications Standards Committee (PASC)

Chair:	Jim Isaak
Vice Chairs:	Hal Jespersen
	Lorraine Kevra
	Pete Meier
	Andrew Twigger
Treasurer	Peter Smith
Secretary:	Shane McCarron

P1224 Working Group Officials

Chair:	Steve Trus
Vice Chair:	Tim Carter
Editor/Secretary:	Graham Jack

Working Group

Greg Bussiere	Mark Karan	Ted Roberts
Tony Cincotta	Eva Kuiper	Enzo Signore
Iain Devine	Edmum Liu	Daniella Sirocchi
Chris Harding	Kevin Montgomery	Rob Spade
Mark Hazzard	Ed Owens	Alec Tashjian
Kheim Ho	Pierre Perrot	Andrew Twigger
Peter Janecek		Bruce Weiner

The following persons were members of the P1224 Balloting Group that approved the standard for submission to the IEEE Standards Board:

Ralph Barker *Uniform Organizational Representative*
Petr Janecek *X/Open Organizational Representative*

David Athersych	Derek Kaufman	Joseph Ramus
Ralph Barker	Norman Kincl	Robert Robillard
Peter Beaman	Joseph King	Wolfgang Schwabl
Andy Bihain	Stephen Layne	Richard Seibel
Mack W. Bishop	Joachim Loitzenbauer	Dan Shia
Tim Carter	Steven Lucas	Jim Sides
Andy Cheese	Joseph F. P. Luhukay	John P. Slone
Kilnam Chon	Roger Martin	James G. Tanner
Ian Devine	Joberto S. B. Martins	Alec Tashjian
Philip H. Enslow	John S. Morris	Ravi Tavakley
K. Frimpong-Ansah	Andrew L. Nicholson	Donn S. Terry
Michel Gien	Jan Osterweil	Trevor J. Thompson
Steve Griesmer	Edward Owens	Mark-Rene Uchida
Scott Guthery	Ramesh Pabbati	Paul Wanish
Charles Hammons	Prayoon Phathayakorn	Bruce Weiner
C. J. Harding	Dave Plauger	Les Wibberley
Mark Hazzard	Udo W. Pooch	Peter Wishart
James Hegerty	Scott E. Preece	Oren Yuen
Niraj Jain	Paul Rabin	George R. Zerdian
Hal Jespersen	Steve Rago	Jason Zions

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

Wallace S. Read, *Chair*
Donald C. Loughry, *Vice Chair*
Andrew G. Salem, *Secretary*

Gilles A. Baril
Clyde R. Camp
Donald C. Fleckenstein
Jay Forster*
David F. Franklin
Ramiro Garcia
Donald N. Heirman
Jim Isaak

Ben C. Johnson
Walter J. Karplus
Lorraine C. Kevra
E. G. "Al" Kiener
Ivor N. Knight
Joseph Koepfinger*
D .N. "Jim" Logothetis
Don T. Michael*

Marco W. Migliaro
L. John Rankine
Arthur K. Reilly
Ronald H. Reimer
Gary S. Robinson
Leonard L. Tripp

*Member Emeritus

Also included are the following nonvoting IEEE Standards Board liaisons:

Satish K. Aggarwal
James Beall

Richard B. Engelman
David E. Soffrin

Stanley I. Warshaw

Rachel Auslander
IEEE Standards Project Editor

CLAUSE	PAGE
1. General	1
1.1 Scope	1
1.2 Normative References	1
1.3 Conformance	2
2. Terminology and General Requirements	2
2.1 Conventions	2
2.2 Definitions	3
3. Test Assertions—General	7
3.1 Scope	7
3.2 Normative References	7
3.3 Conformance	7
3.4 Test Methods	8
4. Test Assertions—Definitions and General Requirements	8
5. Test Assertions—Overview	8
6. Test Assertions—Information Architecture	8
6.1 Introduction	8
6.2 Objects	8
6.3 Object Attributes	8
6.4 Classes	8
6.5 Class Definitions	8
6.6 Packages	9
6.7 Package Definitions	9
6.8 Workspaces	9
6.9 Storage Management	9
7. Test Assertions—Information Syntaxes	9
7.1 Introduction	9
7.2 Syntax Templates	10
7.3 Syntaxes	10
7.4 Strings	10
7.5 Relationship to ASN.1 Simple Types	10
7.6 Relationship to ASN.1 Useful Types	10
7.7 Relationship to ASN.1 Character String Types	10
7.8 Relationship to ASN.1 Type Constructors	10
8. Test Assertions—Service Interface	10
8.1 Introduction	10
8.2 Datatypes	10
8.3 Operations	13
8.4 Return Codes	37

CLAUSE	PAGE
9. Test Assertions—Object Management Package	39
9.1 Class Hierarchy	39
9.2 Class Definitions	39
10. Test Assertions—Definitions of Constants	41
10.1 Introduction	41
10.2 Object Management Package	41
Annex A (Informative) Bibliography	43

IEEE Standard for Information Technology—Test Methods for Measuring Conformance to Open Systems Interconnection (OSI) Abstract Data Manipulation—Application Program Interface (API) [Language Independent]

1. General

1.1 Scope

This standard defines requirements for test methods for measuring conformance to IEEE Std 1224–1993 {2}.

IEEE Std 1224-1993 {2} is stated in terms that are independent of any particular programming language, but each implementation that conforms to it implements a particular programming language binding, and conforms to its programming language binding specification. Each set of test methods for measuring conformance to IEEE Std 1224-1993 {2} assumes a particular programming language binding specification. Each programming language binding specification may impose further programming-language specific requirements on the test methods which, in conjunction with the requirements imposed by this standard, constitute the requirements that shall be satisfied by test methods used for measuring conformance to that programming language binding specification. This standard applies to all sets of test methods for measuring conformance to any programming language binding specification for IEEE Std 1224-1993 {2}.

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}IEEE Std 1003.3-1991, *IEEE Standard for Information Technology—Test Methods for Measuring Conformance to POSIX*.

{2}IEEE Std 1224-1993, *IEEE Standard for Information Technology—Open Systems Interconnection (OSI) Abstract Data Manipulation—Application Program Interface (API) [Language Independent]*.

1.3 Conformance

A set of test methods that conforms to this standard shall conform to IEEE Std 1003.3-1991 {1}, with references to the “POSIX.n test method specification” being interpreted as references to this standard, and references to “the POSIX standard for which conformance is being measured” being interpreted as references to IEEE Std 1224-1993 {2}.

In addition to meeting the conformance criteria defined in IEEE Std 1003.3-1991 {1}, a set of test methods that conforms to this standard shall test all documentation assertions defined in this standard.

NOTE — Conformance to IEEE Std 1003.3-1991 {1} implies that the test methods will test all other assertions defined in this standard.

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. The first letters of language-independent OM class and OM attribute names are capitalized (e.g., **Arbitrary-Encoding**).

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

The use of fonts in this standard is as follows.

- The **Helvetica** font is used for:
 - Language-independent operation names, such as `om_copy_value`
 - Language-independent datatype names, such as `om_exclusions_type`
 - Language-independent error names, such as `no_such_class`
- The *italic* font is used for:
 - Language-independent operation arguments, such as *source_value_position*
 - Introducing important terms
 - Cross-references in Section 2.2
- The **bold** font is used for:
 - Language-independent, concrete OM class names, such as **External**
 - Language-independent OM attribute names, such as **Arbitrary-Encoding**
 - Language-independent OM attribute values, such as **insert-at-certain-point**
- The ***bold italic*** font is used for:
 - Language-independent, abstract OM class names, such as ***Object***
- The **constant width** (Courier) font is used for:
 - References to terms defined in the X.400 standards.

2.1.2 Language-Independent Conventions

The language-independent conventions defined in IEEE Std 1224-1993 {2} shall apply in this standard.

2.1.3 Test Methods Conventions

The assertion classification and labelling schemes defined in IEEE Std 1003.3-1991 {1} are used in this standard, with the following addition.

This standard contains an assertion for each statement that is derived from the standard to which conformance is being measured, that pertains to the documentation associated with the implementation being tested, and that is true for a conforming implementation. These assertions are called documentation assertions. Each documentation assertion is labelled *Dnn*, where *nn* is a two digit number that is unique within this standard.

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 IEEE Std 1224-1993 {2}.

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 [ISO/IEC 9945-1 : 1990 {B1}].

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.

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.
- 2) With respect to applications, an indication of a recommended programming practice for applications and a requirement for Strictly Conforming Application

2.2.1.6 supported: A condition regarding optional functionality.

Certain functionality in this standard is optional, but the interfaces to that functionality are always required. If the functionality is *supported*, the interfaces work as specified by this 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 : 1990 {B1}].

2.2.1.7 system documentation: All documentation provided with an implementation, except the conformance document [ISO/IEC 9945-1 : 1990 {B1}].

Electronically distributed documents for an implementation are considered part of the system documentation.

2.2.1.8 undefined: An indication that this standard imposes no portability requirements on an application's use of an indeterminate value or its behavior with erroneous program constructs or erroneous data [ISO/IEC 9945-1 : 1990 {B1}].

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 IEEE Std 1224-1993 {2}.

2.2.1.9 unspecified: An indication that this standard imposes no portability requirements on applications for correct program constructs or correct data regarding a value or behavior [ISO/IEC 9945-1 : 1990 {B1}].

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 IEEE Std 1224-1993 {2}.

2.2.2 General Terms

For the purposes of this standard, the following definitions apply:

2.2.2.1 abstract class: A class, instances of which are forbidden unless they belong to one of its concrete subclasses [IEEE Std 1224-1993 {2}].

2.2.2.2 accessible object: An object for which the client possesses a valid designator or handle [IEEE Std 1224-1993 {2}].

2.2.2.3 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 [IEEE Std 1224-1993 {2}].

2.2.2.4 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 [IEEE Std 1224-1993 {2}].

2.2.2.5 base assertion: An assertion that is required to be tested for required features and for implemented conditional features [IEEE Std 1003.3-1991 {1}].

2.2.2.6 bit string: An ordered sequence of zero or more bits [IEEE Std 1224-1993 {2}].

2.2.2.7 byte: A unit of machine storage containing an ordered sequence of 8 bits [IEEE Std 1224-1993 {2}].

2.2.2.8 character string: An ordered sequence of zero or more characters [IEEE Std 1224-1993 {2}].

2.2.2.9 class: A category into which objects are placed on the basis of both their purpose and their internal structure [IEEE Std 1224-1993 {2}].

2.2.2.10 client: Software that uses the interface [IEEE Std 1224-1993 {2}].

2.2.2.11 concrete class: A class, instances of which are permitted either by direct instantiation or the instantiation of its concrete subclasses [IEEE Std 1224-1993 {2}].

2.2.2.12 datatype: A collection of distinguished values, together with a collection of characterizing operations on those values [IEEE Std 1224-1993 {2}].

2.2.2.13 documentation assertion: An assertion that pertains to the documentation associated with the implementation being tested.

2.2.2.14 element: Any of the bits of a bit string, the octets of an octet string, or the bytes by means of which the characters of a character string are represented [IEEE Std 1224-1993 {2}].

2.2.2.15 extended assertion: An assertion that is not required to be tested [IEEE Std 1003.3-1991 {1}].

2.2.2.16 generic interface: The interface, defined at a level that is independent of any particular programming language [IEEE Std 1224-1993 {2}].

2.2.2.17 immediate subclass: A subclass, of a class *C*, having no superclasses that are themselves subclasses of *C* [IEEE Std 1224-1993 {2}].

2.2.2.18 immediate subobject: One object that is a value of an attribute of another [IEEE Std 1224-1993 {2}].

2.2.2.19 immediate superclass: The superclass, of a class *C*, having no subclasses that are themselves superclasses of *C* [IEEE Std 1224–1993 {2}].

2.2.2.20 immediate superobject: One object that contains another among its attribute values [IEEE Std 1224–1993 {2}].

2.2.2.21 inaccessible object: An object for which the client does not possess a valid designator or handle [IEEE Std 1224–1993 {2}].

2.2.2.22 independent copy: A copy of the object plus independent copies of all its subobjects (applied recursively) [IEEE Std 1224–1993 {2}].

2.2.2.23 instance: An object in the category represented by a class [IEEE Std 1224–1993 {2}].

2.2.2.24 intermediate datatype: Any of the basic datatypes in terms of which the other, substantive datatypes of the interface are defined [IEEE Std 1224–1993 {2}].

2.2.2.25 length: The number of bytes required to represent any string [IEEE Std 1224–1993 {2}].

2.2.2.26 minimally consistent object: An object that satisfies various conditions set forth in the definition of its class [IEEE Std 1224–1993 {2}].

2.2.2.27 name: An identifier used to represent a specific value of a specific datatype.

This standard does not specify when a name is bound to its value: at compile time, link time, or load time. The use of a name as a parameter in a datatype specification does not imply that its value is defined at compile time [IEEE Std 1224–1993 {2}].

2.2.2.28 object: Any of the complex information objects created, examined, modified, or destroyed by means of the interface [IEEE Std 1224–1993 {2}].

2.2.2.29 octet string: An ordered sequence of zero or more octets [IEEE Std 1224–1993 {2}].

2.2.2.30 operation: A program abstraction with formal input and output parameters that are bound to objects or values in the application that invokes it [IEEE Std 1224–1993 {2}].

2.2.2.31 OSI abstract data manipulation application program interface: The interface defined in [IEEE Std 1224–1993 {2}].

2.2.2.32 package: A group of related classes [IEEE Std 1224–1993 {2}].

2.2.2.33 package closure: 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 [IEEE Std 1224–1993 {2}].

2.2.2.34 position (within a string): The ordinal position of one element of a string relative to another [IEEE Std 1224–1993 {2}].

2.2.2.35 position (within an attribute): The ordinal position of one value relative to another [IEEE Std 1224–1993 {2}].

2.2.2.36 primary representation: The form in which the service supplies an attribute value to the client [IEEE Std 1224–1993 {2}].

2.2.2.37 private object: The internal representation of the language-independent model of an object in a service, and thus unspecified [IEEE Std 1224–1993 {2}].

2.2.2.38 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 [IEEE Std 1224–1993 {2}].

2.2.2.39 public object: The representation by the client of an object in a particular language binding [IEEE Std 1224–1993 {2}].

2.2.2.40 secondary representation: A second form, an alternative to the primary representation, in which the client may supply an attribute value to the service [IEEE Std 1224-1993 {2}].

2.2.2.41 segment: Zero or more contiguous elements of a string [IEEE Std 1224-1993 {2}].

2.2.2.42 service: Software that implements the interface [IEEE Std 1224-1993 {2}].

2.2.2.43 service interface: The interface as realized, for the benefit of the client, by the service as a whole [IEEE Std 1224-1993 {2}].

2.2.2.44 specific: With respect to a class, the attribute types that may appear in an instance of the class but not in an instance of its superclasses [IEEE Std 1224-1993 {2}].

2.2.2.45 status datatype: An abstract datatype whose values may be bound to “control” values as well as “data” values [IEEE Std 1224-1993 {2}].

2.2.2.46 string: An ordered sequence of zero or more bits, octets, or characters, accompanied by the length of the string [IEEE Std 1224-1993 {2}].

2.2.2.47 subclass: One of the classes, designated as such, whose attribute types are a superset of those of another class [IEEE Std 1224-1993 {2}].

2.2.2.48 subobject: An immediate subobject of an object or of one of its subobjects applied recursively [IEEE Std 1224-1993 {2}].

2.2.2.49 superclass: One of the classes, designated as such, whose attribute types are a subset of those of another class [IEEE Std 1224-1993 {2}].

2.2.2.50 superobject: The immediate superobject of an object, or one of its superobjects applied recursively [IEEE Std 1224-1993 {2}].

2.2.2.51 syntax: A category into which an attribute value is placed on the basis of its form [IEEE Std 1224-1993 {2}].

2.2.2.52 syntax template: A lexical construct containing an asterisk from which several attribute syntaxes can be derived by substituting text for the asterisk [IEEE Std 1224-1993 {2}].

2.2.2.53 test methods: The software, procedures, or other means specified to measure conformance to a specification [IEEE Std 1224-1993 {2}].

2.2.2.54 type: A category into which attribute values are placed on the basis of their purpose [IEEE Std 1224-1993 {2}].

2.2.2.55 value: An arbitrarily complex information item that can be viewed as a characteristic or property of an object [IEEE Std 1224-1993 {2}].

2.2.2.56 workspace: A repository for instances of classes in the closures of one or more packages associated with the workspace [IEEE Std 1224-1993 {2}].

2.2.3 Abbreviations

For the purposes of this standard, the following abbreviations apply:

2.2.3.1 API: Application Program Interface

2.2.3.2 ANS.1: Abstract Syntax Notation One (defined in ISO/IEC 8824 :1990 {B3}).

2.2.3.3 BER: Basic Encoding Rules (defined in ISO/IEC 8825 :1990 {B4}).

2.2.3.4 IA5: International Alphabet No. 5 (used in “IA5 String,” see ISO/IEC 8824 :1990 {B3}).

2.2.3.5 OM: OSI abstract data Manipulation.

2.2.3.6 OSI: Open Systems Interconnection.

2.2.3.7 UTC: Universal Coordinated Time.

3. Test Assertions—General

3.1 Scope

There are no test assertions for this clause.

3.2 Normative References

There are no test assertions for this clause.

3.3 Conformance

3.3.1 Implementation Conformance

3.3.1.1 Requirements

- D02 The implementation documentation shall claim to support the version of the interface defined by the current edition of IEEE Std 1224-1993 {2}.
- 001(B) The implementation of the interface shall comprise one or more workspaces.
NOTE — This can only be tested in conjunction with an application-specific API.
- 002(A) Each workspace in the interface shall support the OM package.
- GA02 For op in om_decode:
The op will support use of the BER.
NOTE — There should be a suite of test assertions associated with this. It is beyond the scope of this standard. Refer to ISO/IEC 8824 :1990 {B3} and ISO/IEC 8825 :1990 {B4}.

3.3.1.2 Documentation

3.3.1.3 Conforming Implementation Options

- D03 The local character set representation and the precise mappings between it and the following string syntaxes are documented: **Bit, Encoding, Object Identifier, Octect, General, Generalized Time, Graphic, IA5, Numeric, Object Descriptor, Printable, Teletex, UTC Time, Videotex, Visible.**
- D04 The length of the longest string the om_get operation will return is documented.
- D05 The length of the longest string the om_get operation will return is not less than 1024 octets.
- R01 Whether the service reports an exception if an object supplied to it as an argument is not minimally consistent is documented. (See D06 in 6.5)

3.3.2 Application Conformance

3.3.2.1 Strictly Conforming Application

There are no test assertions for this subclause.

3.3.2.2 Conforming Application

3.3.2.2.1 ISO/IEC Conforming Application

There are no test assertions for this subclause.

3.3.2.3 Conforming Application Using Extensions

There are no test assertions for this subclause.

3.4 Test Methods

There are no test assertions for this clause.

4. Test Assertions—Definitions and General Requirements

There are no test assertions for this section.

5. Test Assertions—Overview

There are no test assertions for this section.

6. Test Assertions—Information Architecture

6.1 Introduction

There are no test assertions for this clause.

6.2 Objects

There are no test assertions for this clause.

6.3 Object Attributes

There are no test assertions for this clause.

6.4 Classes

- R01 The object identifier that denotes a class is an attribute of every instance of the class. See 001(A) in 9.2.1, 005(A) in 9.2.2.].
- R02 Instances of an abstract class are forbidden. (See GA22 in 8.4.)
- R03 When `om_create(abstract_class,...)` is invoked, then the operation returns `not_concrete` error. (See GA22 in 8.4.)
- R04 When `om_create(concrete_class,...)` is invoked and the definition of the concrete class indicates the client may not create instances, then the operation returns `function_declined`. [See 028(A) in 8.3.3.]

6.5 Class Definitions

GA03 For `op` in `om_create`, `om_decode`, `om_encode`, `om_get`: