

INTERNATIONAL STANDARD

ISO/IEC
14362

First edition
1996-06-01

Information technology — Test methods for measuring conformance to Open Systems Interconnection (OSI) abstract data manipulation — Application Program Interface (API) [Language independent]

*Technologies de l'information — Méthodes d'essai pour mesurer la
conformité à la manipulation de données abstraites en interconnexion de
systèmes ouverts (OSI) — Interface de programme d'application (API)
[indépendante du langage]*



Reference number
ISO/IEC 14362:1996(E)

AVA Documents
Document Sharing Hub

Contents

	PAGE
Section 1: General	1
1.1 Scope	1
1.2 Normative References	2
1.3 Conformance	2
Section 2: Terminology and General Requirements	3
2.1 Conventions	3
2.2 Definitions	4
Section 3: Test Assertions—General	11
3.1 Scope	11
3.2 Normative References	11
3.3 Conformance	11
3.4 Test Methods	12
Section 4: Test Assertions—Definitions and General Requirements	13
Section 5: Test Assertions—Overview	15
Section 6: Test Assertions—Information Architecture	17
6.1 Introduction	17
6.2 Objects	17
6.3 Object Attributes	17
6.4 Classes	17
6.5 Class Definitions	18
6.6 Packages	18
6.7 Package Definitions	18
6.8 Workspaces	19
6.9 Storage Management	19
Section 7: Test Assertions—Information Syntaxes	21
7.1 Introduction	21
7.2 Syntax Templates	21
7.3 Syntaxes	21
7.4 Strings	21
7.5 Relationship to ASN.1 Simple Types	22
7.6 Relationship to ASN.1 Useful Types	22
7.7 Relationship to ASN.1 Character String Types	22
7.8 Relationship to ASN.1 Type Constructors	22

© 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

Section 8: Test Assertions—Service Interface	23
8.1 Introduction	23
8.2 Datatypes	23
8.2.1 Signed Integers	23
8.2.2 Unsigned Integers	23
8.2.3 Boolean	23
8.2.4 Enumeration	23
8.2.5 Exclusions	23
8.2.6 Integer	24
8.2.7 Modification	24
8.2.8 Object	24
8.2.9 Object Identifier	24
8.2.10 Private Object	24
8.2.11 Public Object	24
8.2.12 Service-Generated Public Object	24
8.2.13 Client-Generated Public Object	24
8.2.14 Attribute Set	24
8.2.15 Attribute	25
8.2.16 Value	25
8.2.17 Value Data	25
8.2.18 Null Value	25
8.2.19 String Value	25
8.2.20 String Flags	25
8.2.21 String Choice	25
8.2.22 Return Code	25
8.2.23 String	25
8.2.24 Type	26
8.2.25 Type List	26
8.2.26 Value Length	26
8.2.27 Value Position	26
8.2.28 Workspace	26
8.3 Operations	27
8.3.1 Copy	28
8.3.2 Copy Value	33
8.3.3 Create	36
8.3.4 Decode	39
8.3.5 Delete	45
8.3.6 Encode	47
8.3.7 Get	51
8.3.8 Instance	56
8.3.9 Put	58
8.3.10 Read	62
8.3.11 Remove	64
8.3.12 Write	67
8.4 Return Codes	70
Section 9: Test Assertions—Object Management Package	75
9.1 Class Hierarchy	75
9.2 Class Definitions	75
9.2.1 Encoding	75

9.2.2 External(7.3.2) 76

9.2.3 Object 77

Section 10: Test Assertions—Definitions of Constants 79

10.1 Introduction 79

10.2 Object Management Package 79

Annex A (informative) Bibliography 81

Alphabetic Topical Index 83

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 14362 was prepared by IEEE (as IEEE Std 1326-1993) 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 of this International Standard is for information only.

Introduction

(This introduction is not a normative part of ISO/IEC 14362, 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.)

The purpose of this International Standard is to define test methods for measuring conformance to ISO/IEC 14360 {1}, which defines a general purpose OSI Abstract Data Manipulation (OM) Application Program Interface (API) in terms that are independent of any particular programming language.

Related Standards

ISO/IEC 14360 {1} 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 ISO/IEC 14364 {B6}. The requirements specified in this International Standard apply to test methods for measuring conformance to standards defining all such programming language bindings of ISO/IEC 14360 {1}. Additional programming language-specific requirements may be defined for test methods for measuring conformance to each programming language binding of ISO/IEC 14360 {1}. Such a set of requirements, for test methods for measuring conformance to the C programming language binding contained in ISO/IEC 14364 {B6}, is contained in ISO/IEC 14366 {B7}.

ISO/IEC 14360 {1}, 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 ISO/IEC 14361 {B5} and the API to Directory Services defined in ISO/IEC 14392 {B8}. The requirements specified in this International Standard may also apply to test methods for measuring conformance to standards defining such application-specific APIs.

The test assertions in this International Standard were produced in accordance with IEEE Std 1003.3-1991 {2}.

Overview

This International Standard contains a section corresponding to each section of ISO/IEC 14360 {1}. Each such section contains test assertions for that section of ISO/IEC 14360 {1}. Each set of test methods for measuring conformance to ISO/IEC 14360 {1} is required to test these assertions in conformance with the requirements of IEEE Std 1003.3-1991 {2}.

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 International 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.

This International Standard is based on IEEE Std 1326-1993 [B9], 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—Test methods for measuring conformance to Open Systems Interconnection (OSI) abstract data manipulation—Application Program Interface (API) [Language independent]

Section 1: General

1.1 Scope

This International Standard defines requirements for test methods for measuring conformance to ISO/IEC 14360 {1}.

ISO/IEC 14360 {1} 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 ISO/IEC 14360 {1} 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 International Standard, constitute the requirements that shall be satisfied by test methods used for measuring conformance to that programming language binding specification. This International Standard applies to all sets of test methods for measuring conformance to any programming language binding specification for ISO/IEC 14360 {1}.

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/IEC 14360: 1996, *Information technology—Open Systems Interconnection (OSI) abstract data manipulation—Application Program Interface (API) [Language independent]*.
- {2} IEEE Std 1003.3-1991,¹⁾ *IEEE Standard for Information Technology—Test Methods for Measuring Conformance to POSIX*.

1.3 Conformance

A set of test methods that conforms to this International Standard shall conform to IEEE Std 1003.3-1991 {2}, with references to the “POSIX.n test method specification” being interpreted as references to this International Standard, and references to “the POSIX standard for which conformance is being measured” being interpreted as references to ISO/IEC 14360 {1}.

In addition to meeting the conformance criteria defined in IEEE Std 1003.3-1991 {2}, a set of test methods that conforms to this International Standard shall test all documentation assertions defined in this International Standard.

NOTE: Conformance to IEEE Std 1003.3-1991 {2} implies that the test methods will test all other assertions defined in this International Standard.

1) ISO/IEC documents can be obtained from the ISO Central Secretariat, 1 Rue de Varembe, Case Postale 56, CH-1211, Genève 20, Switzerland/Suisse. IEEE documents can be obtained from the IEEE Service Center, 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ, 08855-1331, USA.

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. 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 lower-case and are spelled with underscores between words (e.g., `om_get`).

The use of fonts in this International 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 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 ISO/IEC 14360 {1} shall apply in this International Standard.

2.1.3 Test Methods Conventions

The assertion classification and labelling schemes defined in IEEE Std 1003.3-1991 {2} are used in this International Standard, with the following addition.

This International 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 International Standard.

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 ISO/IEC 14360 {1}.

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 {B1}].

2.2.1.3 may: An indication of an optional feature [ISO/IEC 9945-1 {B1}].

With respect to implementations, the word *may* is to be interpreted as an optional feature that is not required in this International 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 {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 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 {B1}].

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

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

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 [ISO/IEC 9945-1 {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 ISO/IEC 14360 {1}.

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 [ISO/IEC 9945-1 {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 ISO/IEC 14360 {1}.

2.2.2 General Terms

For the purposes of this International 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 [ISO/IEC 14360 {1}].

2.2.2.2 accessible object: An object for which the client possesses a valid designator or handle [ISO/IEC 14360 {1}].

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 [ISO/IEC 14360 {1}].

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 [ISO/IEC 14360 {1}].

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 {2}].

2.2.2.6 bit string: An ordered sequence of zero or more bits [ISO/IEC 14360 {1}].

2.2.2.7 byte: A unit of machine storage containing an ordered sequence of 8 bits [ISO/IEC 14360 {1}].

2.2.2.8 character string: An ordered sequence of zero or more characters [ISO/IEC 14360 {1}].

2.2.2.9 class: A category into which objects are placed on the basis of both their purpose and their internal structure [ISO/IEC 14360 {1}].

2.2.2.10 client: Software that uses the interface [ISO/IEC 14360 {1}].

2.2.2.11 concrete class: A class, instances of which are permitted either by direct instantiation or the instantiation of its concrete subclasses [ISO/IEC 14360 {1}].

2.2.2.12 datatype: A collection of distinguished values, together with a collection of characterizing operations on those values [ISO/IEC 14360 {1}].

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

116 **2.2.2.14 element:** Any of the bits of a bit string, the octets of an octet string, or
117 the bytes by means of which the characters of a character string are represented
[ISO/IEC 14360 {1}].

118 **2.2.2.15 extended assertion:** An assertion that is not required to be tested
119 [IEEE Std 1003.3-1991 {2}].

120 **2.2.2.16 generic interface:** The interface, defined at a level that is independent
121 of any particular programming language [ISO/IEC 14360 {1}].

122 **2.2.2.17 immediate subclass:** A subclass, of a class *C*, having no superclasses
123 that are themselves subclasses of *C* [ISO/IEC 14360 {1}].

124 **2.2.2.18 immediate subobject:** One object that is a value of an attribute of
125 another [ISO/IEC 14360 {1}].

126 **2.2.2.19 immediate superclass:** The superclass, of a class *C*, having no sub-
127 classes that are themselves superclasses of *C* [ISO/IEC 14360 {1}].

128 **2.2.2.20 immediate superobject:** One object that contains another among its
129 attribute values [ISO/IEC 14360 {1}].

130 **2.2.2.21 inaccessible object:** An object for which the client does not possess a
131 valid designator or handle [ISO/IEC 14360 {1}].

132 **2.2.2.22 independent copy:** A copy of the object plus independent copies of all
133 its subobjects (applied recursively) [ISO/IEC 14360 {1}].

134 **2.2.2.23 instance:** An object in the category represented by a class [ISO/IEC
135 14360 {1}].

136 **2.2.2.24 intermediate datatype:** Any of the basic datatypes in terms of which
137 the other, substantive datatypes of the interface are defined [ISO/IEC 14360 {1}].

138 **2.2.2.25 length:** The number of bytes required to represent any string [ISO/IEC
139 14360 {1}].

140 **2.2.2.26 minimally consistent object:** An object that satisfies various condi-
141 tions set forth in the definition of its class [ISO/IEC 14360 {1}].

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

This International 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 [ISO/IEC 14360 {1}].

2.2.2.28 object: Any of the complex information objects created, examined, modified, or destroyed by means of the interface [ISO/IEC 14360 {1}].

2.2.2.29 octet string: An ordered sequence of zero or more octets [ISO/IEC 14360 {1}].

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 [ISO/IEC 14360 {1}].

2.2.2.31 OSI abstract data manipulation application program interface: The interface defined in [ISO/IEC 14360 {1}].

2.2.2.32 package: A group of related classes [ISO/IEC 14360 {1}].

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 [ISO/IEC 14360 {1}].

2.2.2.34 position (within a string): The ordinal position of one element of a string relative to another [ISO/IEC 14360 {1}].

2.2.2.35 position (within an attribute): The ordinal position of one value relative to another [ISO/IEC 14360 {1}].

2.2.2.36 primary representation: The form in which the service supplies an attribute value to the client [ISO/IEC 14360 {1}].

2.2.2.37 private object: The internal representation of the language-independent model of an object in a service, and thus unspecified [ISO/IEC 14360 {1}].

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 [ISO/IEC 14360 {1}].

175 **2.2.2.39 public object:** The representation by the client of an object in a particular language binding [ISO/IEC 14360 {1}].

176 **2.2.2.40 secondary representation:** A second form, an alternative to the pri-
177 mary representation, in which the client may supply an attribute value to the ser-
178 vice [ISO/IEC 14360 {1}].

179 **2.2.2.41 segment:** Zero or more contiguous elements of a string [ISO/IEC 14360
180 {1}].

181 **2.2.2.42 service:** Software that implements the interface [ISO/IEC 14360 {1}].

182 **2.2.2.43 service interface:** The interface as realized, for the benefit of the client,
183 by the service as a whole [ISO/IEC 14360 {1}].

184 **2.2.2.44 specific:** With respect to a class, the attribute types that may appear in
185 an instance of the class but not in an instance of its superclasses [ISO/IEC 14360
186 {1}].

187 **2.2.2.45 status datatype:** An abstract datatype whose values may be bound to
188 “control” values as well as “data” values [ISO/IEC 14360 {1}].

189 **2.2.2.46 string:** An ordered sequence of zero or more bits, octets, or characters,
190 accompanied by the length of the string [ISO/IEC 14360 {1}].

191 **2.2.2.47 subclass:** One of the classes, designated as such, whose attribute types
192 are a superset of those of another class [ISO/IEC 14360 {1}].

193 **2.2.2.48 subobject:** An immediate subobject of an object or of one of its subobjects
194 applied recursively [ISO/IEC 14360 {1}].

195 **2.2.2.49 superclass:** One of the classes, designated as such, whose attribute
196 types are a subset of those of another class [ISO/IEC 14360 {1}].

197 **2.2.2.50 superobject:** The immediate superobject of an object, or one of its
198 superobjects applied recursively [ISO/IEC 14360 {1}].

199 **2.2.2.51 syntax:** A category into which an attribute value is placed on the basis of
200 its form [ISO/IEC 14360 {1}].

201 **2.2.2.52 syntax template:** A lexical construct containing an asterisk from which
202 several attribute syntaxes can be derived by substituting text for the asterisk
203 [ISO/IEC 14360 {1}].

204 **2.2.2.53 test methods:** The software, procedures, or other means specified to
205 measure conformance to a specification [ISO/IEC 14360 {1}].

2.2.2.54 type: A category into which attribute values are placed on the basis of their purpose [ISO/IEC 14360 {1}].

2.2.2.55 value: An arbitrarily complex information item that can be viewed as a characteristic or property of an object [ISO/IEC 14360 {1}].

2.2.2.56 workspace: A repository for instances of classes in the closures of one or more packages associated with the workspace [ISO/IEC 14360 {1}].

2.2.3 Abbreviations

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

2.2.3.1 API: Application Program Interface

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

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

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

Section 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 ISO/IEC 14360 (1).

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 International Standard. Refer to ISO/IEC 8824 (B3) and ISO/IEC 8825 (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, Octet, 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.

1 **Section 4: Test Assertions—Definitions and General Requirements**

2 There are no test assertions for this section.

1 **Section 5: Test Assertions—Overview**

2 There are no test assertions for this section.

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 8.3.3.4.]

6.5 Class Definitions

- GA03 For op in om_create, om_decode, om_encode, om_get:
When op is invoked and completes successfully, then the type of each attribute of any object returned is specific to the object's class or one of its superclasses.
- GA04 For op in om_create, om_decode, om_encode, om_get:
When op is invoked and completes successfully, then the number of each attribute of any object returned is no greater than the class of the object permits.
- GA05 For op in om_create, om_decode, om_encode, om_get:
When op is invoked and completes successfully, then the syntax of each value of any object returned is among those permitted by the class of the object.
- GA06 For op in om_create, om_decode, om_encode, om_get:
When op is invoked and completes successfully, then the number of bits, octets, or characters in each value that is a string in any object returned is among those the class of the object permits.
- R05 For op in om_create, om_decode, om_encode, om_get:
When op is invoked and completes successfully, then any object returned, as a result, is minimally consistent. (See GA03 in 6.5, GA04 in 6.5, GA05 in 6.5, GA06 in 6.5.)
- D06 Whether the service reports an exception if an object supplied as an operation argument is not minimally consistent is documented.
- 001(C) If it is documented that the service reports an exception when an object supplied as an operation argument is not minimally consistent:
When om_put is invoked and an object supplied as an argument is not minimally consistent, then an exception will be reported.

6.6 Packages

There are no test assertions for this clause.

6.7 Package Definitions

There are no test assertions for this clause.

6.8 Workspaces

There are no test assertions for this clause.

6.9 Storage Management

002(B) A private object is destroyed when the workspace containing it is destroyed.

NOTE: It is not expected that this will be tested. This might be tested when this interface is used in conjunction with another API that contains functions for destroying workspaces.

003(B) A service-generated public object is not destroyed when the workspace that generated it is destroyed.

NOTE: It is not expected that this will be tested. This might be tested when this interface is used in conjunction with another API that contains functions for destroying workspaces.

004(B) Handles to private subobjects in a service-generated public object are invalidated when the workspace containing the private subobjects is destroyed.

NOTE: It is not expected that this will be tested. This might be tested when this interface is used in conjunction with another API that contains functions for destroying workspaces.

Section 7: Test Assertions—Information Syntaxes

7.1 Introduction

There are no test assertions for this clause.

7.2 Syntax Templates

There are no test assertions for this clause.

7.3 Syntaxes

There are no test assertions for this clause.

7.4 Strings

001(B) When an Encoding Octet string is created, then the octets are those the BER permit for the contents octets of the encoding of a value of any ASN.1 type.

NOTE: It is not expected that this will be tested.

002(B) When a General, Generalized Time, Graphic, IA5, Numeric, Object Descriptor, Printable, Teletex, Universal, Unrestricted, UTC Time, Videotex or Visible Character string is created, then the characters permitted are those permitted by the ASN.1 type of the same name.

NOTE: It is not expected that this will be tested.

18 003(B) When an Object Identifier Octet string is created, then the octets are
19 those the BER permit for the contents octets of the encoding of a value
20 of the ASN.1 type Object Identifier.

21 NOTE: It is not expected that this will be tested.

22 R01 When a string, other than a local string, is presented at the interface
23 and contains characters that are illegal for its syntax, then the error
24 **wrong-value-syntax** shall be returned. [See GA31 in 8.4.]

25 **7.5 Relationship to ASN.1 Simple Types**

26 There are no test assertions for this clause.

27 **7.6 Relationship to ASN.1 Useful Types**

28 There are no test assertions for this clause.

29 **7.7 Relationship to ASN.1 Character String Types**

30 There are no test assertions for this clause.

31 **7.8 Relationship to ASN.1 Type Constructors**

32 There are no test assertions for this clause.

Section 8: Test Assertions—Service Interface

8.1 Introduction

There are no test assertions for this clause.

8.2 Datatypes

There are no test assertions for this clause.

8.2.1 Signed Integers

There are no test assertions for this subclause.

8.2.2 Unsigned Integers

There are no test assertions for this subclause.

8.2.3 Boolean

There are no test assertions for this subclause.

8.2.4 Enumeration

GA07 For op in om_create, om_decode, om_encode, om_get:

When op is invoked and returns successfully, then any data value returned of the datatype om_enumeration_type is an attribute whose value syntax is an Enumeration.

8.2.5 Exclusions

There are no test assertions for this subclause.