

Recognized as an American National Standard

IEEE Std 1326.2-1993

IEEE Standard for Information Technology—Test Methods for Measuring Conformance to Directory Services—Application Program Interface (API) [Language Independent]

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: Test methods for IEEE Std 1224.2-1993 language-independent specification of the applications interface to directory services are defined.

Keywords: API, application portability, open systems, directory, portable application

The Institute of Electrical and Electronics Engineers, Inc.
345 East 47th Street, New York, NY 10017-2394, USA

Copyright © 1994 by the
Institute of Electrical and Electronics Engineers, Inc.
All rights reserved, Published 1994
Printed in the United States of America

ISBN 1-55937-310-5

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 may involve the use of patented technology. Their approval by the Institute of Electrical and Electronics Engineers, Inc. does not mean that using such technology for the purpose of conforming to such standards is authorized by the patent owner. It is the obligation of the user of such technology to obtain all necessary permissions.

Introduction

(This introduction is not a normative part of IEEE Std 1326.2-1993, IEEE Standard for Information Technology—Test Methods for Measuring Conformance to Directory Services—Application Program Interface (API) [Language Independent], but is included for information only.)

Purpose

The purpose of this standard is to define test methods for the language-independent specification contained in IEEE Std 1224.2-1993 {14} of the application programming interface (API) to directory services.

A directory is a distributed collection of information that programs can access in order to make queries or updates. IEEE Std 1224.2-1993 {14} defines, in programming language-independent terms, an API to directory services. This API is known as the Directory Services API (DS API).

The DS API is intended to be used to provide access to a range of directory services that are instances of a common abstract model. That model is defined in the CCITT X.500 Series recommendations and in ISO/IEC 9594 : 1990. IEEE Std 1224.2-1993 {14} prescribes how the DS API is to be used to access the particular directory service defined in ISO/IEC 9594 : 1990 and indicates how it may be used to access other directory services that conform to the same abstract model. Nothing in IEEE Std 1224.2-1993 {14} or in this standard requires that the implementation of the interface or the Directory itself actually make use of the Directory Access Protocol (DAP), the Directory System Protocol (DSP) or other parts of the model, just so long as it provides the defined service.

The interface is designed for operational interactions with a directory, rather than for management interactions such as knowledge management or schema management. Also, security features are not generally visible in the interface, in order to permit flexibility in security policies. It is intended that an application program should be able to use the interface to access a single directory service or to access several directory services at the same time.

Related Standards

IEEE Std 1224.2-1993 {14} is intended to provide the basis for the definition of programming language bindings to which implementations and applications can conform. A specification for such a language binding, for the C programming language, is contained in IEEE Std 1327.2-1993 {B7}. This standard applies to test methods for measuring conformance to any such programming language binding specification.

The API defined in IEEE Std 1224.2-1993 {14} uses the mechanism for OSI abstract data Manipulation (OM) defined in IEEE Std 1224-1993 {13}. IEEE Std 1326-1993 {B5} defines the requirements that shall apply to test methods for measuring conformance to IEEE Std 1224-1993 {13}.

IEEE Std 1003.3-1991 {12} defines the general requirements that shall apply to test methods for measuring conformance to POSIX. A set of test methods used to measure conformance to IEEE Std 1224.2-1993 {14} must satisfy the requirements of IEEE Std 1003.3-1991 {12}, as well as conforming to this standard.

Overview

For each section of IEEE Standard 1224.2-1993 {14}, this standard contains a corresponding section containing test assertions, in accordance with IEEE Std 1003.3-1991 {12}. A set of test methods conforms to this standard if it tests all of the test assertions.

Acknowledgements

The base document for this standard is an unpublished set of test assertions for the X/Open Directory Services API Specification (XDS) {B10}, copyright © 1991-2 by X/Open Company Limited, United Kingdom. The working group is grateful to X/Open for permission to use their materials.

This standard was prepared by the Namespace and Directory Services Working Group (P1224.2, formerly P1003.17), sponsored by the Portable Applications Standards Committee of the IEEE Computer Society. At the time this standard was approved, the membership of the P1224.2 working group was as follows:

Portable Application Standards Committee (PASC)

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

P1224.2 Working Group Officials

Chair:	Mark Hazzard
Vice chair:	Donna Fisher
Editor:	Chris Harding
Secretary:	Donna Fisher

P1224.2 Working Group

Greg Bussiere	Edmun Liu	Rob Spade
Scott Guthery	Ted Roberts	Alec Tashjian

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

Stephen R. Walli *EurOpen Organizational Representative*
Ralph Barker *Uniform Organizational Representative*

Reed M. Adams	Steve Griesmer	Ronald K. Sellars
Charles R. Arnold	Charles Hammons	Rob Spade
Timothy Baker	Michael Hannah	Sandra Swearingen
Ralph Barker	Chris Harding	James G. Tanner
Andy Bihain	Mark Hazzard	Alec Tashjian
Mack W. Bishop	Barry Hedquist	Donn S. Terry
Robert Bismuth	Khiem Ho	Alan E. Turner
Kevin Brady	Petr Janeczek	Andrew T. Twigger
Dawn Burnett	Hal Jespersen	Mark-Rene Uchida
Gregory Bussiere	Robert Knighten	Andrew Walker
George Carson	Kevin Lewis	Paul Wanish
Andy Cheese	Roger Martin	Peter Wishart
Fred D. Crouner	Mark Modig	Oren Yuen
Ana Maria De Alvare	Diana Norwood	John Zenor
Stephen L. Diamond	Edward Owens	George R. Zerdian
Philip H. Enslow	Scott E. Preece	Jason Zions
Donna K. Fisher	Paul Rabin	John Zolnowsky
Donald Folland	Seth Rosenthal	
Michel Gien	Richard Seibel	

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	Ben C. Johnson	Marco W. Migliaro
Clyde R. Camp	Walter J. Karplus	L. John Rankine
Donald C. Fleckenstein	Lorraine C. Kevra	Arthur K. Reilly
Jay Forster*	E. G. "Al" Kiener	Ronald H. Reimer
David F. Franklin	Ivor N. Knight	Gary S. Robinson
Ramiro Garcia	Joseph Koepfinger*	Leonard L. Tripp
Donald N. Heirman	D. N. "Jim" Logothetis	Donald W. Zipse
Jim Isaak	Don T. Michael*	

*Member Emeritus

Also included are the following nonvoting IEEE Standards Board liaisons:

Satish K. Aggarwal	Richard B. Engelman	Stanley I. Warshaw
James Beall	David E. Soffrin	
	Rachel A. Meisel	
	<i>IEEE Standards Project Editor</i>	

CLAUSE	PAGE
1. General	1
1.1 Scope	1
1.2 Normative References	1
1.3 Conformance	2
2. Terminology and General Requirements	3
2.1 Conventions	3
2.2 Definitions	4
3. Test Assertions—General	11
3.1 Scope	11
3.2 Normative References	11
3.3 Conformance	11
4. Test Assertions—Terminology and General Requirements	12
5. Test Assertions—Description	12
5.1 General	12
5.2 Initialization and Shutdown	12
5.3 Session	12
5.4 Context	12
5.5 Operation Parameters	13
5.6 Operation Results	14
5.7 Synchronous and Asynchronous Operations	15
5.8 Automatic Connection Management	17
5.9 Automatic Continuation and Referral Handling	17
6. Test Assertions—Interface Operations	17
6.1 Datatype Definitions	17
6.2 Abandon	17
6.3 Add-Entry	20
6.4 Bind	27
6.5 Compare	30
6.6 Initialize	39
6.7 List	39
6.8 Modify-Entry	48
6.9 Modify-RDN	56
6.10 Read	64
6.11 Receive-Result	72
6.12 Remove-Entry	75
6.13 Search	82
6.14 Shutdown	93
6.15 Unbind	93
6.16 Version	95

CLAUSE	PAGE
7. Test Assertions—Interface Class Definitions	96
7.1 Overview	96
7.2 Class Hierarchy	97
7.3 Access Point	97
7.4 Address	97
7.5 Attribute	97
7.6 Attribute List	98
7.7 AVA	98
7.8 Common Results	98
7.9 Compare Result	99
7.10 Context	100
7.11 Continuation Reference	103
7.12 DS DN	104
7.13 DS RDN	104
7.14 Entry Information	104
7.15 Entry Information Selection	105
7.16 Entry Modification	106
7.17 Entry Modification List	107
7.18 Extension	107
7.19 Filter	107
7.20 Filter Item	108
7.21 List Info	109
7.22 List Info Item	110
7.23 List Result	110
7.24 Name	111
7.25 Operation Progress	111
7.26 Partial Outcome Qualifier	112
7.27 Presentation Address	113
7.28 Read Result	113
7.29 Relative Name	114
7.30 Search Info	114
7.31 Search Result	114
7.32 Session	115
8. Test assertions—Error Class Definitions	116
8.1 Introduction	116
8.2 OM Class Hierarchy	117
8.3 Error	117
8.4 Abandon Failed	117
8.5 Attribute Error	118
8.6 Attribute Problem	118
8.7 Communications Error	120
8.8 Library Error	120
8.9 Name Error	122
8.10 Referral	123
8.11 Security Error	123
8.12 Service Error	124
8.13 System Error	126
8.14 Update Error	127

CLAUSE	PAGE
9. Test assertions—The X.500 Directory Service.....	128
9.1 Introduction.....	129
9.2 Package Object Identifiers	129
9.3 Selected Attribute Types	130
9.4 Selected Object Classes	136
9.5 OM Class Hierarchy.....	136
9.6 Algorithm Identifier	136
9.7 Certificate.....	136
9.8 Certificate List.....	137
9.9 Certificate Pair	137
9.10 Certificate Sublist.....	138
9.11 Certificates	138
9.12 Cross Certificates	138
9.13 DL Submit Permission	139
9.14 Facsimile Telephone Number	139
9.15 Forward Certification Path.....	139
9.16 Postal Address.....	139
9.17 Search Criterion	140
9.18 Search Guide	140
9.19 Signature	140
9.20 Teletex Terminal Identifier	141
9.21 Telex Number	141
10. Test Assertions—Constant Definitions.....	141
10.1 Introduction.....	141
10.2 Directory Service Package	141
10.3 Basic Directory Contents Package.....	144
10.4 Strong Authentication Package.....	145
10.5 MHS Directory User Package.....	146
Annex A (Informative) Bibliography	147

IEEE Standard for Information Technology—Test Methods for Measuring Conformance to Directory Services—Application Program Interface (API) [Language Independent]

1. General

1.1 Scope

This standard defines requirements for test methods for measuring conformance to IEEE Std 1224.2-1993 {14}.

IEEE Std 1224.2-1993 {14} 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 1224.2-1993 {14} assumes a particular programming language binding specification. Each programming language binding specification may impose further, programming language specific, requirements on the test methods that, 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 1224.2-1993 {14}.

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 8824:1990 (CCITT Recommendation X.208: 1988),¹ *Information technology—Open Systems Interconnection—Specification of Abstract Syntax Notation One (ASN.1)*

{2}ISO/IEC 8825:1990 (CCITT Recommendation X.209: 1988), *Information technology—Open Systems Interconnection—Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1)*.

{3}ISO/IEC 9072-1:1989 (CCITT Recommendation X.219: 1988), *Information processing systems—Open Systems Interconnection—Text Communication—Remote Operations—Part 1: Model, Notation and Service Definition*.

{4}ISO/IEC 9072-2: 1989 (CCITT Recommendation X.229: 1988), *Information processing systems—Open Systems Interconnection—Text Communication—Remote Operations—Part 2: Protocol Specification*.

{5}ISO/IEC 9594-1:1990 (CCITT Recommendation X.500: 1988), *Information technology—Open Systems Interconnection—The Directory—Part 1: Overview of concepts, models and services*.

{6}ISO/IEC 9594-2: 1990 (CCITT Recommendation X.501: 1988), *Information technology—Open Systems Interconnection—The Directory—Part 2: Models*.

{7}ISO/IEC 9594-3: 1990 (CCITT Recommendation X.511: 1988), *Information technology—Open Systems Interconnection—The Directory—Part 3: Abstract service definition*.

{8}ISO/IEC 9594-4: 1990 (CCITT Recommendation X.518: 1988), *Information technology—Open Systems Interconnection—The Directory—Part 4: Procedures for distributed operation*.

{9}ISO/IEC 9594-6: 1990 (CCITT Recommendation X.520: 1988), *Information technology—Open Systems Interconnection—The Directory—Part 6: Selected attribute types*.

{10}ISO/IEC 9594-7: 1990 (CCITT Recommendation X.521: 1988), *Information technology—Open Systems Interconnection—The Directory—Part 7: Selected object classes*.

{11}ISO/IEC 9594-8: 1990 (CCITT Recommendation X.509: 1988), *Information technology—Open Systems Interconnection—The Directory Part 8: Authentication framework*.

{12}IEEE Std 1003.3-1991,² *IEEE Standard for Information Technology—Test Methods for Measuring Conformance to POSIX*.

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

{14}IEEE Std 1224.2-1993, *IEEE Standard for Information Technology—Directory Services—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 {12}, with references in IEEE Std 1003.3-1991 {12} to the “POSIX.n test method specification” being interpreted as references to this standard, and references in IEEE Std 1003.3-1991 {12} to “the POSIX standard for which conformance is being measured” being interpreted as references to IEEE Std 1224.2-1993 {14}.

¹ISO and ISO/IEC documents can be obtained from the ISO office, 1 Rue de Varembe, Case Postale 56, CH-1211, Genève 20, Switzerland/Suisse. IEEE publications are available from the Institute of Electrical and Electronics Engineers, Service Center, 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855-1331, USA.

²IEEE documents are available from the Institute of Electrical and Electronics Engineers, Service Center, 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855-1331, USA.

In addition to meeting the conformance criteria defined in IEEE Std 1003.3-1991 {12}, 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 {12} 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

Many technical terms, such as “object” and “attribute”, are used in describing both OSI abstract data manipulation and directory services. In the first case they refer to constructs of the OSI abstract data manipulation interface, while in the second case they refer to constructs of the directory service to which the interface provides access. The meanings ascribed to these terms in these cases are often similar but different. Throughout this standard, care is taken to distinguish between these meanings, for example by qualifying the term with “OM” or “directory” as appropriate (as with OM classes and directory classes and with OM attributes and directory attributes). The unqualified term “attribute” denotes the directory construct, while the phrase “OM attribute” denotes the OSI abstract data manipulation construct. The phrase “Object Class” denotes the directory construct, while the phrase “OM class” denotes the OSI abstract data manipulation construct.

The term “operation” is used in two different senses, by the language independent specification conventions and by the directory service descriptions. When used in the sense of the Directory, it is always qualified by the term “directory”, in the phrase “directory operation.” The unqualified term “operation” denotes the language-independent specification construct. Sometimes, to reinforce the distinction, the phrase “interface operation” is used to denote the language-independent specification construct, instead of “operation.”

The reader is urged to be cautious, and reference to 2.2 (Definitions) may be useful.

Language-independent concrete OM class names, OM attribute names, and OM attribute value names appear in bold font while abstract OM class names appear in bold italic font. They are all spelled with hyphens between words. The first letters of language-independent OM class and OM attribute names are capitalized (e.g., **Filter-Item**).

Language-independent datatype, operation argument, and error names appear in Helvetica font, are lowercased, and are spelled with underscores between words (e.g., `ds_abandon`).

Each OM class has an associated datatype, whose name is obtained from the class name by converting uppercase to lowercase, converting hyphens to underscores, adding the prefix “ds_”, and adding the suffix “_type”. (Thus, `ds_filter_item_type` is the datatype corresponding to OM class **Filter-Item**.)

When important terms are introduced, they appear in italics. Italics are also used in 2.2 for cross-references.

The use of fonts in this standard is as follows:

- 1) The Helvetica font is used for
 - Language-independent operation names, such as `ds_add_entry`
 - Language-independent datatype names, such as `ds_status_type`
 - Language-independent error names, such as `Abandon-Failed`
- 2) The *italic* font is used for
 - Language-independent operation arguments, such as *session*

- The introduction of important terms
- Cross-references in 2.2
- 3) The **bold** font is used for
 - Language-independent concrete OM class names, such as **Filter-Item**
 - Language-independent OM attribute names, such as **Filter-Item-Type**
 - Language-independent OM attribute values, such as **approximate-match**
- 4) The **bold italic** font is used for
 - Language-independent abstract OM class names, such as *Common-Results*
- 5) The constant width (Courier) font is used for
 - Terms defined in standards upon which this standard is based

2.1.2 Representation of Strings of Numbers

Strings of numbers (such as ASN.1 Object Identifier encodings) are represented in the form “\xaa\xbb\xcc..”, where each term \xnn represents an element of the string, and the value of that element expressed in hexadecimal notation is nn. So, for example, the string consisting of the three numbers 1, 10, 100 is represented as “\x01\x0a\x64”.

2.1.3 Language-Independent Conventions

The language-independent conventions assume a model of an abstract OM object class mechanism and an abstract procedural interface, although programming language bindings may take other forms. The language-independent specification conventions are described in IEEE Std 1224-1993 {13}.

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 of 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 locally administered: An indication that the configuration is not consistent across all systems and that the administrator of each system shall provide documentation of its behavior. [IEEE Std 1224.2-1993 {14}]

2.2.1.4 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 that can be provided. With respect to Strictly Conforming Applications, the word *may* means that the optional feature shall not be used. [ISO/IEC 9945-1 : 1990 {B1}]

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

2.2.1.6 should:

- 1) With respect to implementations, an indication of an implementation recommendation but not a requirement
- 2) With respect to applications, an indication of a recommended programming practice for applications and a requirement for Strictly Conforming Applications. [ISO/IEC 9945-1 :1990 {B1}]

2.2.1.7 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 in 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}]

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

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

2.2.1.9 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.

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

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

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

2.2.2 General Terms

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

2.2.2.1 abstract class: An OM class of OM objects of which instances are forbidden. [IEEE Std 1224.2-1993 {14}]

2.2.2.2 Abstract Syntax Notation One (ASN.1): A notation that both enables complicated types to be defined and also enables values of these types to be specified. See ISO/IEC 8824 :1990 {1}. [IEEE Std 1224.2-1993 {14}]

2.2.2.3 access point: The point at which an abstract service is obtained. [ISO/IEC 9594-2 : 1990 {6}]

2.2.2.4 address: An unambiguous name, label, or number that identifies the location of a particular entity or service. [IEEE Std 1224.2-1993 {14}]

See also *presentation address* in 2.2.2.70.

2.2.2.5 alias [alias name]: An alternate name for a directory object, provided by the use of one or more alias entries in the DIT. [ISO/IEC 9594-2 :1990 {6}]

2.2.2.6 alias entry: A directory entry, of object class “alias,” containing information used to provide an alternative name for an object. [IEEE Std 1224.2-1993 {14}]

2.2.2.7 alias name: See *alias* in 2.2.2.5.

2.2.2.8 argument: Information that is passed to an interface operation or a directory operation. [IEEE Std 1224.2-1993 {14}]

2.2.2.9 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 {13}]

2.2.2.10 asynchronous completion: Completion of an asynchronous directory operation.

An asynchronous directory operation is complete when a corresponding synchronous directory operation would complete and any associated status fields have been updated. [IEEE Std 1224.2-1993 {14}]

2.2.2.11 asynchronous operation: An operation that does not of itself cause the process requesting the operation to be blocked from further use of the CPU.

This implies that the process and the operation are running concurrently. [IEEE Std 1224.2-1993 {14}]

2.2.2.12 attribute: Used alone, means *directory attribute* as in 2.2.2.30 (but see also *OM attribute* in 2.2.2.61).

2.2.2.13 attribute syntax [directory syntax]: A definition of the set of values that a directory attribute may assume.

It includes the datatype, in ASN.1, and, usually, one or more matching rules by which values may be compared. [IEEE Std 1224.2-1993 {14}]

2.2.2.14 attribute type: Used alone, means *directory attribute type* as in 2.2.2.31 (see also *OM attribute type* in 2.2.2.62).

2.2.2.15 attribute value: Used alone, means *directory attribute value* as in 2.2.2.32 (see also *OM attribute value* in 2.2.2.63).

2.2.2.16 attribute value assertion: A proposition, which may be true, false, or undefined, about the values (or perhaps only the distinguished values) of an entry. [ISO/IEC 9594-2 :1990 {6}]

NOTE — An attribute value assertion consists of an attribute type and a single value. Loosely, it is true if one of the values of the given attribute in the entry matches the given value.

2.2.2.17 attribute value syntax: See *OM syntax* in 2.2.2.66.

2.2.2.18 base assertion: An assertion that is required to be tested for required features and for implemented conditional features. [IEEE Std 1003.3-1991 {12}]

2.2.2.19 basic encoding rules: A specific set of rules used to encode ASN. 1 values as strings of octets. [IEEE Std 1224.2-1993 {14}]

2.2.2.20 cache: See *copy* in 2.2.2.25.

2.2.2.21 certificate: A synonym for *user certificate*, 2.2.2.89.

2.2.2.22 chaining: A mode of interaction optionally used by a DSA that cannot perform a directory operation itself; the DSA chains by invoking a directory operation of another DSA and then relaying the outcome to the original requester. [ISO/IEC 9594-4 :1990 {8}]

2.2.2.23 class: See *object class* in 2.2.2.58 and *OM class* in 2.2.2.64.

2.2.2.24 continuation reference: A reference that describes how the performance of all or part of a directory operation can be continued at a different DSA or DSAs. [IEEE Std 1224.2-1993 {14}]

See also *referral* in 2.2.2.75.

2.2.2.25 copy:

- a) A copy of an entry stored in other DSA(s) through bilateral agreement.
- b) A locally and dynamically stored copy of an entry resulting from a request (a cache copy). [IEEE Std 1224.2-1993 {14}]

2.2.2.26 concrete class: An OM class of which instances are permitted. [IEEE Std 1224.2-1993 {14}]

2.2.2.27 descriptor: The means by which the client and service exchange an OM attribute value, and the integers that denote its representation, type, and syntax. [IEEE Std 1224.2-1993 {14}]

2.2.2.28 descriptor list: An ordered sequence of descriptors that is used to represent several OM attribute types and values. [IEEE Std 1224.2-1993 {14}]

2.2.2.29 directory: A repository of information about objects that provides directory services to its users, which allows its users access to the information. [ISO/IEC 9594-2 : 1990 {6}]

2.2.2.30 directory attribute [attribute]: The information of a particular type concerning an object and appearing in an entry describing the object in the DIB. [ISO/IEC 9594-2 : 1990 {6}]

2.2.2.31 directory attribute type [attribute type]: That component of a *directory attribute* that indicates the class of information given by that attribute.

It is an *object identifier*, and so completely unique. [ISO/IEC 9594-2 : 1990 {6}]

2.2.2.32 directory attribute value [attribute value]: A particular instance of the class of information indicated by a directory attribute type. [ISO/IEC 9594-2 : 1990 {6}]

2.2.2.33 directory class: See *object class* in 2.2.2.58.

2.2.2.34 directory information base: The complete set of information to which the directory provides access and that includes all of the pieces of information that can be read or manipulated using the operations of the directory. [ISO/IEC 9594-2 : 1990 {6}]

2.2.2.35 directory information tree: The directory information base, considered as a tree, whose vertices (other than the root) are the directory entries. [ISO/IEC 9594-2 : 1990 {6}]

2.2.2.36 directory object: Anything in some “world,” generally the world of telecommunications and information processing or some part thereof, that is identifiable (can be named) and that it is of interest to hold information on in the directory information base. [ISO/IEC 9594-2 : 1990 {6}]

2.2.2.37 directory operation: Processing performed within the directory to provide a service, such as a read directory operation; it is given some arguments as input, performs some processing, and returns some results.

An application process causes a directory operation to be performed by invoking an interface operation. [IEEE Std 1224.2-1993 {14}]

2.2.2.38 directory syntax: See *attribute syntax* in 2.2.2.13.

2.2.2.39 directory system agent: An OSI application process that is part of the Directory. [ISO/IEC 9594-2 : 1990 {6}]

2.2.2.40 directory user agent: An OSI application process that represents a user accessing the Directory. [ISO/IEC 9594-2 : 1990 {6}]

NOTE — This may be composed of an arbitrary number of system processes and application processes, including the one or more that are the user(s).

2.2.2.41 distinguished encoding: Restrictions to the basic encoding rules designed to ensure a unique encoding of each ASN.1 value, defined in clause 8.7 of ISO/IEC 9594-8 : 1990 {11}. [IEEE Std 1224.2-1993 {14}]

2.2.2.42 distinguished name: One of the names of an object, formed from the sequence of RDNs of its object entry and each of its superior entries. [ISO/IEC 9594-2 : 1990 {6}]

2.2.2.43 distinguished value: An attribute value in an entry that has been designated to appear in the relative distinguished name of the entry. [IEEE Std 1224.2-1993 {14}]

2.2.2.44 documentation assertion: An assertion that pertains to the documentation associated with the implementation being tested. [IEEE Std 1326-1993 {B5}]

2.2.2.45 entry [object entry]: A part of the DIB that contains information about an object.

Each entry is made up of attributes. [ISO/IEC 9594-2 : 1990 {6}]

2.2.2.46 extended assertion: An assertion that is not required to be tested. [IEEE Std 1003.3-1991 {12}]

2.2.2.47 filter: An assertion about the presence or value of certain attributes of an entry in order to limit the scope of a search. [ISO/IEC 9594-3 : 1990 {7}]

2.2.2.48 function: A programming language construct, modeled after the mathematical concept.

A function encapsulates some behavior. It is given zero or more arguments as input, performs some processing, and returns some results. Functions are also known as procedures, subprograms, or subroutines. See also *operation*, 2.2.2.67. [IEEE Std 1224.2-1993 {14}]

2.2.2.49 immediate subordinate: An entry in the DIT that is immediately below another entry in the tree.

The distinguished name of the immediate subordinate is formed by appending its relative distinguished name to the distinguished name of the other entry. [IEEE Std 1224.2-1993 {14}]

2.2.2.50 immediate superior: An entry in the DIT that is immediately above another entry in the tree.

The distinguished name of the immediate superior, followed by the relative distinguished name of the other entry, forms the distinguished name of the other entry.

Each entry (except the root) has exactly one immediate superior. [IEEE Std 1224.2-1993 {14}]

2.2.2.51 interface operation: See *operation* in 2.2.2.67.

2.2.2.52 invoke ID: An identifier used to distinguish one directory operation from all other outstanding operations. [IEEE Std 1224.2-1993 {14}]

2.2.2.53 knowledge reference: Knowledge that associates, either directly or indirectly, a DIT entry with the DSA in which it is located. [ISO/IEC 9594-4 : 1990 {8}]

2.2.2.54 leaf entry: A directory entry that has no subordinates.

It can be an alias entry or an object entry. [IEEE Std 1224.2-1993 {14}]

2.2.2.55 name: A construct that singles out a particular (directory) object from all other objects.

A name shall be unambiguous (that is, denote just one object); however, it need not be unique (that is, be the only name that unambiguously denotes the object). [ISO/IEC 9594-2 : 1990 {6}]

2.2.2.56 nonspecific subordinate reference: A knowledge reference that holds information about a DSA that holds one or more unspecified subordinate entries. [ISO/IEC 9594-4 : 1990 {8}]

2.2.2.57 object: See *directory object* in 2.2.2.36 and *OM object* in 2.2.2.65.

2.2.2.58 object class [directory class]: An identified family of objects (or conceivable objects) that share certain characteristics. [IEEE Std 1224.2-1993 {14}]

2.2.2.59 object entry: See *entry* in 2.2.2.45.

2.2.2.60 object identifier: A value (distinguishable from all other such values) that is associated with an information object. See ISO/IEC 8824 : 1990 {1}. [IEEE Std 1224.2-1993 {14}]

2.2.2.61 OM attribute: A component of an OM 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.

2.2.2.62 OM attribute type: A category into which OM attribute values are placed on the basis of their purpose.

2.2.2.63 OM attribute value: An arbitrarily complex information item that can be viewed as a characteristic or property of an OM object.

2.2.2.64 OM class: A category into which OM objects are placed on the basis of both their purpose and their internal structure.

2.2.2.65 OM object: Any of the complex information objects created, examined, modified, or destroyed by means of the OM interface.

2.2.2.66 OM syntax [attribute value syntax]: A category into which an OM attribute value is placed on the basis of its form.

2.2.2.67 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 {13}]

2.2.2.68 outstanding directory operation: A directory operation, invoked asynchronously (i.e., with **asynchronous=true** in the context), that has not yet been the subject of an invocation of the **ds_abandon** interface operation or the **ds_receive_results** interface operation. [IEEE Std 1224.2-1993 {14}]

2.2.2.69 package: A group of related OM classes.

2.2.2.70 presentation address: An unambiguous name that is used to identify a set of presentation service access points.

Loosely, it is the network address of an OSI service. [IEEE Std 1224.2-1993 {14}]

2.2.2.71 private OM object: The internal representation of the language-independent model of an *OM object* in a service, and thus unspecified. [IEEE Std 1224-1993 {13}]

2.2.2.72 process: An address space, a single thread of control that executes within that address space, and its required system resources.

On a system that implements *threads*, a process is redefined to consist of an address space with one or more threads executing within that address space and their required system resources. [ISO/IEC 9945-1 : 1990 {B1}]

NOTE — The term *process* is used in contrast to “system process,” or the OSI usage of the term “application process.”

2.2.2.73 public OM Object: The representation by the client of an object in a particular programming language binding. [IEEE Std 1224-1993 {13}]

2.2.2.74 purported name: A construct that is syntactically a name but that has not (yet) been shown to be a valid name. [ISO/IEC 9594-2 : 1990 {6}]

2.2.2.75 referral: An outcome that can be returned by a DSA that cannot perform a directory operation itself, and that identifies one or more other DSAs more able to perform the directory operation. [ISO/IEC 9594-4 : 1990 {8}]

2.2.2.76 relative distinguished name: A set of AVAs, each of which is true, concerning the distinguished values of a particular entry. [ISO/IEC 9594-2 : 1990 {6}]

2.2.2.77 replication: The process by which copies of entries are created and maintained. [IEEE Std 1224.2-1993 {14}]

2.2.2.78 result: Information that is returned from an interface operation or a directory operation and that constitutes the outcome of the processing that was performed. [IEEE Std 1224.2-1993 {14}]

2.2.2.79 schema: The set of rules and constraints concerning DIT structure, object class definitions, attribute types, and syntaxes that characterize the DIB. [ISO/IEC 9594-2 : 1990 {6}]

2.2.2.80 service controls: Parameters conveyed as part of a directory operation that constrain various aspects of its performance. [ISO/IEC 9594-3 : 1990 {7}]

2.2.2.81 session: A sequence of directory operations requested by a particular user of a particular DUA using the same session OM object. [IEEE Std 1224.2-1993 {14}]

2.2.2.82 signed: The condition of information that has an enciphered summary appended to it that is used to ensure the integrity of the data, the authenticity of the originator, and the unambiguous relationship between the originator and the data. [IEEE Std 1224.2-1993 {14}]

2.2.2.83 subordinate: The converse of superior. [ISO/IEC 9594-2 : 1990 {6}]

2.2.2.84 superior (applying to entry or directory object): Immediately superior, or superior to one that is immediately superior (recursively). [ISO/IEC 9594-2 : 1990 {6}]

2.2.2.85 syntax: See *attribute syntax* in 2.2.2.13 and *OM syntax* in 2.2.2.66.

2.2.2.86 test methods: The software, procedures, or other means specified to measure conformance to a specification. [IEEE Std 1224-1993 {13}]

2.2.2.87 thread: A single sequential flow of control within a process. [IEEE Std 1224.2-1993 {14}]

2.2.2.88 user: The entity or person that accesses the Directory, used in this standard to refer to the application program that is using the interface. [IEEE Std 1224.2-1993 {14}]

2.2.2.89 user certificate [certificate]: The public keys of a user, together with some other information, rendered unforgeable by encipherment with the secret key of the certification authority that issued it. [ISO/IEC 9594-8 : 1990 {11}]

2.2.2.90 value: See *attribute value* in 2.2.2.15 and *OM attribute value* in 2.2.2.63.

2.2.2.91 workspace: A repository for instances of OM classes in the closures of one or more packages associated with the workspace. [IEEE Std 1224-1993 {13}]

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 ASN.1: Abstract Syntax Notation One.

2.2.3.3 AVA: Attribute Value Assertion.

2.2.3.4 BDCP: Basic Directory Contents Package.

2.2.3.5 BER: Basic Encoding Rules.

2.2.3.6 CA: Certification Authority.

2.2.3.7 CCITT: International Telegraph and Telephone Consultative Committee.

2.2.3.8 DAP: Directory Access Protocol.

2.2.3.9 DIB: Directory Information Base.

2.2.3.10 DIT: Directory Information Tree.

2.2.3.11 DL: Delivery List.

2.2.3.12 DMD: Directory Management Domain.

2.2.3.13 DN: Distinguished Name.

2.2.3.14 DS: Directory Service.

2.2.3.15 DS API: Directory Services API (as defined in IEEE Std 1224.2-1993 {14} and IEEE Std 1327.2-1993 {B7}).

2.2.3.16 DS Package: Directory Service Package.

2.2.3.17 DSA: Directory System Agent.

2.2.3.18 DSP: Directory System Protocol.

2.2.3.19 DUA: Directory User Agent.

2.2.3.20 IA5: International Alphabet No. 5.

2.2.3.21 ID: Identifier.

2.2.3.22 Interface: The interface to directory services defined in IEEE Std 1224.2-1993 {14} and IEEE Std 1327.2-1993 {B7}.

2.2.3.23 ISDN: Integrated Services Digital Network.

2.2.3.24 ISO: International Organisation for Standardization.

2.2.3.25 MDUP: MHS Directory User Package.

2.2.3.26 MHS: message Handling System (as in ISO/IEC 10021 : 1990 {B2}).

2.2.3.27 MS: Message Store.

2.2.3.28 NBP: Nonbasic Parameter.

2.2.3.29 O/R: Originator/Recipient.

2.2.3.30 OM: OSI abstract data Manipulation.

2.2.3.31 OSI: Open Systems Interconnection.

2.2.3.32 POSIX.3: IEEE Std 1003.3-1991 {12}.

2.2.3.33 RDN: Relative Distinguished Name.

2.2.3.34 ROSE: Remote Operations Service Element (see ISO/IEC 9072 :1989 and CCITT Recommendation X.219:1988 {3} and {4}).

2.2.3.35 X.500 directory standards: ISO/IEC 9954 : 1990 and the 1988 CCITT X.500 Series Recommendations ({5}–{11}).

2.2.3.36 SAP: Strong Authentication Package.

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

There are no test assertions for this subclause.

3.3.1.2 Documentation

There are no test assertions for this subclause.

3.3.2 Conforming Implementation Options

- GAO1 For `op` in `ds_abandon`, `ds_add_entry`, `ds_bind`, `ds_compare`, `ds_initialize`, `ds_list`, `ds_modify_entry`, `ds_modify_rdn`, `ds_receive_result`, `ds_read`, `ds_remove_entry`, `ds_search`, `ds_shutdown`, `ds_unbind`, `ds_version`:
`op` exists in the implementation.
- D01 The maximum number of outstanding asynchronous directory operations is documented.
- D02 Whether or not an asynchronous interface operation returns control before the directory operation is submitted to the directory is documented.
- D03 The manner in which the retrieval of completed directory operations by `ds_receive_result` is prioritized is documented.
- D04 The text and language of error messages is documented.
- D05 Whether or not `ds_abandon` issues an `Abandon` directory operation is documented.
- D06 The OM classes permitted as values of the *Name* argument to interface operations are documented.

3.3.3 Application Conformance

There are no test assertions for this section.

4. Test Assertions—Terminology and General Requirements

There are no test assertions for this section.

5. Test Assertions—Description

5.1 General

- GA02 For `op` in `ds_abandon`, `ds_add_entry`, `ds_compare`, `ds_list`, `ds_modify_entry`, `ds_modify_rdn`, `ds_read`, `ds_remove_entry`, `ds_search`:
The Abstract Service defined in the standards (with the same name as `op`) maps to `op`.
- GA03 For `op` in `ds_bind`, `ds_unbind`:
The Abstract Services `DirectoryBind` and `DirectoryUnbind` defined in the standards map to `op`, taking the values `ds_bind` and `ds_unbind`, respectively.

5.2 Initialization and Shutdown

- 001(A) When `ds_initialize` is invoked and completes successfully, then it returns a workspace that supports the standard Directory Service package.
- 002(B) When `ds_initialize` is invoked and completes successfully, then it returns a workspace that supports no packages other than the standard Directory Service package and no extensions that affect the behavior of applications that make use only of the standard features.
- GA04 For `op` in `ds_compare`, `ds_list`, `ds_receive_result`, `ds_read`, `ds_search`:
When the workspace returned by `ds_initialize` (and not subsequently extended) is used by `up` then any directory attribute values returned by `up` will be represented by OM objects of class **Encoding**.
- R01 The implementation allows for sections of code to invoke `ds_version`, `ds_initialize`, `ds_shutdown` independently, using distinct workspaces. [See 002(A) in 6.6.2.]
- 003(A) When `ds_version` is invoked, supplied with OSI Object Identifiers to identify additional packages or other extensions and those packages or extensions are supported by the implementation, then they are incorporated in the workspace.

5.3 Session

- R01 When `ds_bind` is invoked, passing **Default-Session** as the Session argument then a session with default parameters will be started. [See 003(A) in 6.4.3.]
- 001(B) When `ds_shutdown` is invoked and completes successfully, it will have the effect of deleting any private objects or service-generated public objects that are in the workspace.
- 002(A) The interface supports multiple concurrent sessions.

5.4 Context

- GA05 For `op` in `ds_add_entry`, `ds_compare`, `ds_list`, `ds_modify_entry`, `ds_modify_rdn`, `ds_read`, `ds_remove_entry`, `ds_search`:
When `op` is invoked, then each of the `ServiceControls` defined in the Abstract Service maps onto an OM attribute in the **Context** that is used.

- GA06 For *op* in *ds_add_entry*, *ds_compare*, *ds_list*, *ds_modify_entry*, *ds_modify_rdn*, *ds_read*, *ds_remove_entry*, *ds_search*:
When *op* is invoked, then the value of each component of the **Context** is determined and remains fixed throughout the directory operation.
- GA07 For *op* in *ds_add_entry*, *ds_compare*, *ds_list*, *ds_modify_entry*, *ds_modify_rdn*, *ds_read*, *ds_remove_entry*, *ds_search*:
When *op* is invoked, passing **Default-Context** as the *Context* argument, then it has the same effect as passing the **Context** object with default values.

5.5 Operation Parameters

- GA08 For *op* in *ds_abandon*, *ds_add_entry*, *ds_compare*, *ds_list*, *ds_modify_entry*, *ds_modify_rdn*, *ds_read*, *ds_remove_entry*, *ds_search*:
The Abstract service defined in the standards (with the same name as *op*) has associated arguments that map to the corresponding arguments for *op*.
- GA09 For *op* in *ds_bind*, *ds_unbind*:
The Abstract Services **DirectoryBind** and **DirectoryUnbind** defined in the standards has associated arguments that map to the corresponding arguments for *op*, taking the values *ds_bind* and *ds_unbind*, respectively.
- GA10 For *op* in *ds_abandon*, *ds_add_entry*, *ds_bind*, *ds_compare*, *ds_list*, *ds_modify_entry*, *ds_modify_rdn*, *ds_read*, *ds_remove_entry*, *ds_search*, *ds_unbind*:
When *op* is specified to accept an instance of a particular OM class as the value of an argument, then it will also accept an instance of any subclass of the OM class.
- GA11 For *op* in *ds_abandon*, *ds_add_entry*, *ds_bind*, *ds_compare*, *ds_list*, *ds_modify_entry*, *ds_modify_rdn*, *ds_receive_result*, *ds_read*, *ds_remove_entry*, *ds_search*, *ds_unbind*:
If the implementation recognizes a subclass of a class required in an argument (*new_class*) that is an extension:
When *op*(. *new_class*..) is invoked, then *new_class* will be recognized.
- GA12 For *op* in *ds_abandon*, *ds_add_entry*, *ds_bind*, *ds_compare*, *ds_list*, *ds_modify_entry*, *ds_modify_rdn*, *ds_receive_result*, *ds_read*, *ds_remove_entry*, *ds_search*, *ds_unbind*:
If the implementation does not recognize a subclass of a class required in an argument (*new_class*) that is an extension:
When *op*(. *new_class*..) is invoked, then it will ignore all OM attributes in *new_class* that are not permitted in the required OM class.
- R01 The interface will always return private objects. (See GA21 in 5.6.)

5.5.1 Attribute and AVA

- GA13 If the implementation supports the Basic Directory Contents Package:
For *op* in *ds_compare*, *ds_list*, *ds_receive_result*, *ds_read*, *ds_search*:
When the use of the BDCP has been negotiated and *op* is invoked and completes successfully, returning any attribute values that are defined in the BDCP, then they are represented as specified in the BDCP.
- GA14 If the implementation supports the Strong Authentication Package:
For *op* in *ds_compare*, *ds_list*, *ds_receive_result*, *ds_read*, *ds_search*:
When the use of the SAP has been negotiated and *op* is invoked and completes successfully, returning any attribute values that are defined in the SAP, then they are represented as specified in the SAP.
- GA15 For *op* in *ds_compare*, *ds_list*, *ds_receive_result*, *ds_read*, *ds_search*:
When *op* is invoked and completes successfully, returning any attribute values that are of ASN.1 simple type and are not known, then they are represented as shown in the table below.

Type	Syntax
Bit String	String(Bit)
Boolean	Boolean
Integer	Integer
Null	Null
Object Identifier	String(Object Identifier)

- GA16 For *op* in *ds_compare*, *ds_list*, *ds_receive_result*, *ds_read*, *ds_search*:
When *op* is invoked and completes successfully, returning any attribute values whose attribute type is not known that is an ASN.1 structured type, then they are represented by OM objects of class **Encoding**.
- R01 The OM class **AVA** is a subclass of OM class **Attribute** constrained to have exactly one value. The OM attributes of the instance of **AVA** that is returned are as stated in R01 in 7.7. (See GA40 in 7.7.)
- R02 The OM object of OM class **Attribute-List** contains a single, multi-valued OM attribute whose values are OM objects of OM class **Attribute**. (See R01 in 7.6.)

5.5.2 Entry Information Selection

- GA17 For *op* in *ds_read*, *ds_search*:
When *op* is invoked and completes successfully, then attribute types will always be returned.
- GA18 For *op* in *ds_read*, *ds_search*:
When *op*(.. *selection*..) is invoked and completes successfully and the constant **Select-All-Types** is passed instead of *selection* then just the types of all attributes will be returned.
- GA19 For *op* in *ds_read*, *ds_search*:
When *op*(.. *selection*..) is invoked and completes successfully and the constant **Select-All-Types-And-Values** is passed instead of *selection* then the types and values of all attributes will be returned.

5.5.3 Name

- R01 For *op* in *ds_compare*, *ds_list*, *ds_read*, *ds_search*:
When *op*(.. *context*.. *name*..) is invoked and **Dont-Dereference-Aliases** in *context* is not set, then any aliases in *name* will be dereferenced. (See GA42 in 7.10.)
- R02 For *op* in *ds_compare*, *ds_list*, *ds_read*, *ds_search*:
When *op*(.. *context*.. *name*..) is invoked and **Dont-Dereference-Aliases** in *context* is set, then any aliases in *name* will not be dereferenced. (See GA42 in 7.10.)
- GA20 For *op* in *ds_add_entry*, *ds_modify_entry*, *ds_modify_rdn*, *ds_remove_entry*:

When *op* is invoked, then any aliases in the name will not be dereferenced.

5.6 Operation Results

- R01 For *op* in *ds_abandon*, *ds_add_entry*, *ds_bind*, *ds_compare*, *ds_list*, *ds_modify_entry*, *ds_modify_rdn*, *ds_receive_result*, *ds_read*, *ds_remove_entry*, *ds_search*, *ds_unbind*, *ds_version*:
When *op* is invoked and completes successfully, then a status result is returned. (See GA24 in 5.6.3.)
- GA21 For *op* in *ds_bind*, *ds_compare*, *ds_list*, *ds_receive_result*, *ds_read*, *ds_search*:
When *op* is invoked and the associated directory operation completes successfully, then *op* will return private objects in the workspace associated with the directory session in the invocation of *ds_bind*.

5.6.1 The Invoke ID Parameter

- GA22 If the implementation supports asynchronous operation:
 For `op` in `ds_add_entry`, `ds_compare`, `ds_list`, `ds_modify_entry`, `ds_modify_rdn`, `ds_receive_result`, `ds_read`, `ds_remove_entry`, `ds_search`:
 When `op` is invoked and asynchronous operation is selected and the associated directory operation completes successfully, then the numerical value returned is guaranteed to be unique amongst all outstanding directory operations in a given session. The value is such as could be returned from the Remote Operations Service Element (ROSE) defined in ISO/IEC 9072 :1989 ({3} and {4}).

5.6.2 Result

- GA23 For `op` in `ds_compare`, `ds_list`, `ds_receive_result`, `ds_read`, `ds_search`:
 When `op` is invoked and the associated directory operation does not complete successfully, then a result is not returned.
- R01 For `op` in `ds_compare`, `ds_list`, `ds_receive`, `ds_read`, `ds_search`:
 When `op` is invoked and the associated directory operation does not complete successfully, then the errors returned are reported in the status result. (See, GA24 in 5.6.3.)
- R02 If the implementation supports asynchronous operation:
 For `op` in `ds_add_entry`, `ds_compare`, `ds_list`, `ds_modify_entry`, `ds_modify_rdn`, `ds_search`, `ds_read`, `ds_remove_entry`:
 When `op` is invoked and asynchronous operation is selected, then the result of the associated directory operation is returned by a later invocation of `ds_receive_result`. (See GA28 in 5.7.)

5.6.3 Status

- GA24 For `op` in `ds_abandon`, `ds_add_entry`, `ds_bind`, `ds_compare`, `ds_initialize`, `ds_list`, `ds_modify_entry`, `ds_modify_rdn`, `ds_receive_result`, `ds_read`, `ds_remove_entry`, `ds_search`, `ds_shutdown`, `ds_unbind`, `ds_version`:
 When `op` is invoked, then the status result that is returned is one of the following: the constant **success**, the constant **no_workspace**, the constant **invalid_workspace**, an attribute error represented by a private object of class **Attribute-Error**, a referral represented by a private object of class **Referral**, or an error represented by a private object of one of the subclasses of class **Error**.

5.7 Synchronous and Asynchronous Operations

- R01 If the implementation supports asynchronous operation:
 For `op` in `ds_add_entry`, `ds_compare`, `ds_list`, `ds_modify_entry`, `ds_modify_rdn`, `ds_search`, `ds_read`, `ds_remove_entry`:
 When the value of the **Asynchronous** OM attribute in the *Context* passed to `op` is **true** then the directory operation is asynchronous. The OM attributes of the instance of **Context** that is returned are as stated in R01 in 7.10. (See GA56 in 7.10.)
- R02 If the implementation supports asynchronous operation:
 For `op` in `ds_add_entry`, `ds_compare`, `ds_list`, `ds_modify_entry`, `ds_modify_rdn`, `ds_search`, `ds_read`, `ds_remove_entry`:
 When the value of the **Asynchronous** OM attribute in the *Context* passed to `op` is **false** then the directory operation is synchronous. The OM attributes of the instance of **Context** that is returned are as stated in R01 in 7.10. (See GA55 in 7.10.)
- GA25 If the implementation does not support asynchronous operation:
 For `op` in `ds_add_entry`, `ds_compare`, `ds_list`, `ds_modify_entry`, `ds_modify_rdn`, `ds_search`, `ds_read`, `ds_remove_entry`:
 The directory operation associated with `op` is synchronous.

- GA26 If the implementation supports asynchronous operation:
For *op* in *ds_add_entry*, *ds_compare*, *ds_list*, *ds_modify_entry*, *ds_modify_rdn*, *ds_search*, *ds_read*, *ds_remove_entry*:
When the value of the **Asynchronous** OM attribute is not specified in the *Context* passed to *op* then *op* behaves as if the value of the OM attribute was **false** and the directory operation is synchronous.
- GA27 If the implementation supports asynchronous operation:
For *op* in *ds_add_entry*, *ds_compare*, *ds_list*, *ds_modify_entry*, *ds_modify_rdn*, *ds_search*, *ds_read*, *ds_remove_entry*:
When *op* is invoked and asynchronous operation is selected and *op* completes successfully, then the *invoke_id* of the directory operation is returned by *op*.
- GA28 If the implementation supports asynchronous operation:
For *op* in *ds_add_entry*, *ds_compare*, *ds_list*, *ds_modify_entry*, *ds_modify_rdn*, *ds_search*, *ds_read*, *ds_remove_entry*:
When *op* is invoked and asynchronous operation is selected and completes successfully, then the *invoke_id* returned uniquely identifies the pending result and can be passed to *ds_abandon* to cause the directory operation to be abandoned, or passed to *ds_receive_result* to receive the results for that directory operation.
- GA29 For *op* in *ds_add_entry*, *ds_compare*, *ds_list*, *ds_modify_entry*, *ds_modify_rdn*, *ds_search*, *ds_read*, *ds_remove_entry*:
If the implementation defines that all interface operations return control after an asynchronous directory operation has been submitted to the underlying directory service:
When *op* is invoked and asynchronous operation is selected, then it returns control after waiting for the associated directory operation to be submitted to the underlying directory service.
Otherwise:
When *op* is invoked and asynchronous operation is selected, then it returns control as soon as possible.
- D07 The limit on the number of asynchronous directory operations that may be outstanding at any one time in any one session is documented.
- 001(C) If the implementation supports asynchronous operation:
The constant **Max-Outstanding-Operations** is defined as a value of at least one.
- 002(C) If the implementation does not support asynchronous operation:
The constant **Max-Outstanding-Operations** is defined with the value zero.
- R03 The value of **Max-Outstanding-Operations** indicates the presence or absence of asynchronous operation.
[See both 001(C) in 5.7 and 002(C) in 5.7.]
- GA30 If the implementation supports asynchronous operation:
For *op* in *ds_add_entry*, *ds_compare*, *ds_list*, *ds_modify_entry*, *ds_modify_rdn*, *ds_search*, *ds_read*, *ds_remove_entry*:
When *op* is invoked and asynchronous operation is selected and the number of outstanding directory operations is equal to **Max-Outstanding-Operations** then it returns an instance of class **Library-Error** with **too-many-operations** value of the **Problem** attribute.
- GA31 If the implementation supports asynchronous operation:
For *op* in *ds_add_entry*, *ds_compare*, *ds_list*, *ds_modify_entry*, *ds_modify_rdn*, *ds_search*, *ds_read*, *ds_remove_entry*:
When *op* is invoked and asynchronous operation is selected and an error is detected before the associated asynchronous request is submitted to the directory, then *op* returns control immediately and no outstanding directory operation is generated.
- GA32 If the implementation supports asynchronous operation:
For *op* in *ds_add_entry*, *ds_compare*, *ds_list*, *ds_modify_entry*, *ds_modify_rdn*, *ds_search*, *ds_read*, *ds_remove_entry*:
When *op* is invoked and asynchronous operation is selected and an error occurs after the associated asynchronous request is submitted to the directory, then errors are reported by *ds_receive_result*.

5.8 Automatic Connection Management

There are no test assertions for this clause.

5.9 Automatic Continuation and Referral Handling

- R01 For *op* in *ds_add_entry*, *ds_compare*, *ds_list*, *ds_modify_entry*, *ds_modify_rdn*, *ds_read*, *ds_remove_entry*, *ds_search*:
When continuation references are received during the execution of *op* then the implementation will use them to attempt to contact the other DSA(s) whenever practical. (See GA58 in 7.10.)
- GA33 For *op* in *ds_add_entry*, *ds_compare*, *ds_list*, *ds_modify_entry*, *ds_modify_rdn*, *ds_read*, *ds_remove_entry*, *ds_search*:
When continuation references have all been used by *op* to attempt to contact other DSA(s), then *op* returns a result or error.
- R02 For *op* in *ds_add_entry*, *ds_compare*, *ds_list*, *ds_modify_entry*, *ds_modify_rdn*, *ds_read*, *ds_remove_entry*, *ds_search*:
When **Automatic-Continuations** is **false** in the **Context** object and *op* is invoked, then *op* returns all the continuation references. (See GA57 in 7.10.)

6. Test Assertions—Interface Operations

NOTE — There are a number of instantiations in this section that come from general assertions in section 5. that are conditionally extended assertions. These assertions can only be reliably tested by examining the protocol going between the DUA and the DSA (DAP) or between the DSAs (DSP) which is not a required feature on all implementations. On systems that support the X.500 protocols, these assertions can be tested portably. The assertions are therefore classified as being base assertions for this class of systems and extended assertions for other systems. This applies to all the assertions relating to the **Context** object that are followed by a testing requirements statement explaining how to test the assertion by examining the protocol.

There are other assertions that cannot be tested portably because they require the implementation to force the production of a referral (i.e., assertions that involve the generation of **Access-Point**, **Presentation-Address**, **Continuation-Reference**, **Operation-Progress**, **Partial-Outcome-Qualifier**). Such assertions are marked as being extended, unconditionally.

6.1 Datatype Definitions

There are no test assertions for this clause.

6.2 Abandon

6.2.1 Synopsis

- 001(A) *ds_abandon* exists in the implementation. (See GA01 in 3.3.2.)
- 002({X.500_PROTOCOLS}?A:B)The Abstract Service Abandon defined in the standards maps to *ds_abandon*. (See GA02 in 5.1.)

6.2.2 Description

- 001(C) If the implementation supports asynchronous operation:
When *ds_abandon*(, *invoke_id*) is invoked and completes successfully, then the result of the outstanding asynchronous directory operation identified by *invoke_id* is abandoned.

- 002(D) If the implementation supports asynchronous operation:
When `ds_abandon(, invoke_id)` is invoked and completes successfully, then the result of the outstanding asynchronous directory operation identified by `invoke_id` is never returned by `ds_receive_result`.

6.2.3 Arguments In

- 001(D) If the implementation recognizes a subclass of a class required in an argument (*new_class*) that is an extension:
When `ds_abandon(.. new_class..)` is invoked, then *new_class* will be recognized. (See GA11 in 5.5.)
- 002(C) If the implementation does not recognize a subclass of a class required in an argument (*new_class*) that is an extension:
When `ds_abandon(.. new_class..)` is invoked, then it will ignore all OM attributes in *new_class* that are not permitted in the required OM class. (See GA12 in 5.5.)
- 003({X.500_PROTOCOLS}?A:B)The Abstract Service Abandon defined in the standards has associated arguments that map to the corresponding arguments for `ds_abandon`. (See GA08 in 5.5.)
- 004(A) When `ds_abandon` is specified to accept an instance of a particular OM class as the value of an argument, then it will also accept an instance of any subclass of the OM class. (See GA10 in 5.5.)

6.2.4 Results

- 001(A) When `ds_abandon` is invoked, then the status result that is returned is one of the following: the constant **success**, the constant **no_workspace**, the constant **invalid_workspace**, an attribute error represented by a private object of class **Attribute-Error**, a referral represented by a private object of class **Referral**, or an error represented by a private object of one of the subclasses of class **Error**. (See GA24 in 5.6.3.)
- 002(A) When `ds_abandon` is invoked and completes successfully, then the returned value of type `ds_result_type` will be **success**. (See GA92 in 8.1.)
- 003(A) When `ds_abandon` is invoked and does not complete successfully and has been given a *session* argument containing a valid workspace, then it returns a private object of class **Error**, **Attribute-Error**, or **Referral** in that workspace. (See GA93 in 8.1.)

6.2.5 Errors

- 001(A) When `ds_abandon` is invoked with a *session* argument that does not contain a valid workspace, then **no_workspace** is returned. (See GA95 in 8.1.)
- 002(C) If the implementation supports asynchronous operation:
When `ds_abandon` is invoked and an attempt was made to abandon the directory operation `ds_add_entry` for which this is prohibited, then an **Abandon-Failed** error is returned with the value of the attribute **Problem** set to **cannot-abandon**. The OM attributes of the instance of **Abandon-Failed** that is returned are as stated in R01 in 8.4. (See GA100 in 8.4.)
- 003(C) If the implementation supports asynchronous operation:
When `ds_abandon` is invoked and an attempt was made to abandon directory operation `ds_compare` that could not be abandoned, then an **Abandon-Failed** error is returned with the value of the **Problem** attribute set to **cannot-abandon**. The OM attributes of the instance of **Abandon-Failed** that is returned are as stated in R01 in 8.4. (See GA101 in 8.4.)
- 004(C) If the implementation supports asynchronous operation:
When `ds_abandon` is invoked and an attempt was made to abandon directory operation `ds_list` that could not be abandoned, then an **Abandon-Failed** error is returned with the value of the **Problem** attribute set to **cannot-abandon**. The OM attributes of the instance of **Abandon-Failed** that is returned are as stated in R01 in 8.4. (See GA101 in 8.4.)
- 005(C) If the implementation supports asynchronous operation:
When `ds_abandon` is invoked and an attempt was made to abandon the directory operation `ds_modify_entry` for which this is prohibited, then an **Abandon-Failed** error is returned with the value of the attribute **Problem** set to **cannot-abandon**. The OM attributes of the instance of **Abandon-Failed** that is returned are as stated in R01 in 8.4. (See GA100 in 8.4.)

- 006(C) If the implementation supports asynchronous operation:
When `ds_abandon` is invoked and an attempt was made to abandon the directory operation `ds_modify_rdn` for which this is prohibited, then an **Abandon-Failed** error is returned with the value of the attribute **Problem** set to **cannot-abandon**. The OM attributes of the instance of **Abandon-Failed** that is returned are as stated in R01 in 8.4. (See GA100 in 8.4.)
- 007(C) If the implementation supports asynchronous operation:
When `ds_abandon` is invoked and an attempt was made to abandon directory operation `ds_read` that could not be abandoned, then an **Abandon-Failed** error is returned with the value of the **Problem** attribute set to **cannot-abandon**. The OM attributes of the instance of **Abandon-Failed** that is returned are as stated in R01 in 8.4. (See GA101 in 8.4.)
- 008(C) If the implementation supports asynchronous operation:
When `ds_abandon` is invoked and an attempt was made to abandon the directory operation `ds_remove_entry` for which this is prohibited, then an **Abandon-Failed** error is returned with the value of the attribute **Problem** set to **cannot-abandon**. The OM attributes of the instance of **Abandon-Failed** that is returned are as stated in R01 in 8.4. (See GA100 in 8.4.)
- 009(C) If the implementation supports asynchronous operation:
When `ds_abandon` is invoked and an attempt was made to abandon directory operation `ds_search` that could not be abandoned, then an **Abandon-Failed** error is returned with the value of the **Problem** attribute set to **cannot-abandon**. The OM attributes of the instance of **Abandon-Failed** that is returned are as stated in R01 in 8.4. (See GA101 in 8.4.)
- 010({X.500_PROTOCOLS}?A:B)When `ds_abandon` is invoked and a communications error occurs (including those arising in Remote Operation, Association Control, Presentation, Session, and Transport), then a **Communications-Error** is returned with the value of the **Problem** attribute set to **communications-problem**. The OM attributes of the instance of **Communications-Error** that is returned are as stated in R01 in 8.7. (See GA117 in 8.7.)
- 011(A) When `ds_abandon` is invoked and the OM class of an argument is not supported for this directory operation, then a **Library-Error** is returned with the value of the **Problem** attribute set to **bad-class**. The OM attributes of the instance of **Library-Error** that is returned are as stated in R01 in 8.8.8. (See GA120 in 8.8.)
- 012(A) When `ds_abandon(session,)` is invoked and an invalid session was supplied, then a **Library-Error** is returned with the value of the **Problem** attribute set to **bad-session**. The OM attributes of the instance of **Library-Error** that is returned are as stated in R01 in 8.8. (See GA123 in 8.8.)
- 013(B) When `ds_abandon` is invoked and the interface cannot clear a transient system error by retrying the affected system call, then a **Library-Error** is returned with the value of the **Problem** attribute set to **miscellaneous**. The OM attributes of the instance of **Library-Error** that is returned are as stated in R01 in 8.8. (See GA125 in 8.8.)
- 014(C) If the implementation supports asynchronous operation:
When `ds_abandon` is invoked and an attempt to start a synchronous directory operation was made while there were outstanding asynchronous directory operations, then a **Library-Error** is returned with the value of the **Problem** attribute set to **mixed-synchronous**. The OM attributes of the instance of **Library-Error** that is returned are as stated in R01 in 8.8. (See GA127 in 8.8.)
- 015(A) When `ds_abandon` is invoked and an attempt was made to use optional functionality that is not available in the implementation, then a **Library-Error** is returned with the value of the **Problem** attribute set to **not-supported**. The OM attributes of the instance of **Library-Error** that is returned are as stated in R01 in 8.8. (See GA128 in 8.8.)
- 016(B) When a system error occurs and `ds_abandon` generates an instance of OM class **System-Error**, then its **Problem** attribute will identify the cause of the failure. The OM attributes of the instance of **System-Error** that is returned are as stated in R01 in 8.13. (See GA154 in 8.13.)
- 017(B) When `ds_abandon` is invoked and a transient error occurs, then the affected directory operation will be retried and these errors will not be returned. (See GA155 in 8.13.)

6.2.6 Cross-References

There are no test assertions for this subclause.

6.3 Add-Entry

6.3.1 Synopsis

001(A) `ds_add_entry` exists in the implementation. (See GA01 in 3.3.2.)

6.3.2 Description

- 001(A) When `ds_add_entry(,, entry,)` is invoked, where `entry` is an object that conforms to the directory schema, and the associated directory operation completes successfully, then a leaf entry is added to the directory.
- 002(A) When `ds_add_entry(,, entry,)` is invoked, where `entry` is an alias that conforms to the directory schema, and the associated directory operation completes successfully, then a leaf entry is added to the directory.
- 003({X.500_PROTOCOLS}?A:B)The Abstract Service `AddEntry` defined in the standards maps to `ds_add_entry`. (See GA02 in 5.1.)
- 004(C) If the implementation does not support asynchronous operation:
The directory operation associated with `ds_add_entry` is synchronous. (See GA25 in 5.7.)
- 005(D) If the implementation defines that all interface operations return control after an asynchronous directory operation has been submitted to the underlying directory service:
When `ds_add_entry` is invoked and asynchronous operation is selected, then it returns control after waiting for the associated directory operation to be submitted to the underlying directory service.
Otherwise:
When `ds_add_entry` is invoked and asynchronous operation is selected, then it returns control as soon as possible. (See GA29 in 5.7.)
- 006(C) If the implementation supports asynchronous operation:
When `ds_add_entry` is invoked and asynchronous operation is selected and an error is detected before the associated asynchronous request is submitted to the directory, then `ds_add_entry` returns control immediately and no outstanding directory operation is generated. (See GA31 in 5.7.)
- 007(C) If the implementation supports asynchronous operation:
When `ds_add_entry` is invoked and asynchronous operation is selected and an error occurs after the associated asynchronous request is submitted to the directory, then errors are reported by `ds_receive_result`. (See GA32 in 5.7.)
- 008(A) When continuation references have all been used by `ds_add_entry` to attempt to contact other DSAs, then `ds_add_entry` returns a result or error. (See GA33 in 5.9.)

6.3.3 Arguments In

- 001(D) If the implementation recognizes a subclass of a class required in an argument (*new_class*) that is an extension:
When `ds_add_entry(.. new_class..)` is invoked, then *new_class* will be recognized. (See GA11 in 5.5.)
- 002(C) If the implementation does not recognize a subclass of a class required in an argument (*new_class*) that is an extension:
When `ds_add_entry(.. new_class..)` is invoked, then it will ignore all OM attributes in *new_class* that are not permitted in the required OM class. (See GA12 in 5.5.)
- 003(B) When `ds_add_entry` is invoked, then the `ServiceControls` defined in the Abstract Service each map onto an OM attribute in the **Context** that is used. (See GA05 in 5.4.)
- 004(A) When `ds_add_entry` is invoked, then the value of each component of the **Context** is determined and remains fixed throughout the directory operation. (See GA06 in 5.4.)
- 005(A) When `ds_add_entry` is invoked, passing **Default-Context** as the *Context* argument, then it has the same effect as passing the **Context** object with default values. (See GA07 in 5.4.)
- 006({X.500_PROTOCOLS}?A:B)The Abstract Service `AddEntry` defined in the standards has associated arguments that map to the corresponding arguments for `ds_add_entry`. (See GA08 in 5.5.)
- 007(A) When `ds_add_entry` is specified to accept an instance of a particular OM class as the value of an argument, then it will also accept an instance of any subclass of the OM class. (See GA10 in 5.5.)

- 008(A) When `ds_add_entry` is invoked, then any aliases in the name will not be dereferenced. (See GA20 in 5.5.3.)
- 009(C) If the implementation supports asynchronous operation:
When the value of the **Asynchronous** OM attribute is not specified in the *Context* passed to `ds_add_entry`, then `ds_add_entry` behaves as if the value of the OM attribute was **false** and the directory operation is synchronous. (See GA26 in 5.7.)
- 010({X.500_PROTOCOLS}?A:B)When `ds_add_entry(, context,,)` is invoked and **Chaining-Prohibited** is set, then chaining, and other methods of distributing the request around the directory are prohibited. (See GA41 in 7.10.)
Testing Requirements: In a distributed system this assertion can be tested by checking the Directory Access Protocol or the Directory System Protocol.
- 011({X.500_PROTOCOLS}?A:B)When `ds_add_entry(, context,,)` is invoked and **Dont-Use-Copy** is set, then the request is to be satisfied only by access to directory entries and not copies maintained in other DSAs or locally cached copies. (See GA43 in 7.10.)
Testing Requirements: In a distributed system this assertion can be tested by checking the Directory Access Protocol or the Directory System Protocol.
- 012({X.500_PROTOCOLS}?A:B)When `ds_add_entry(, context,,)` is invoked and **Local-Scope** is set, then the request is to be satisfied locally, the meaning of which is configured by an administrator. (See GA44 in 7.10.)
Testing Requirements: In a distributed system this assertion can be tested by checking the Directory Access Protocol or the Directory System Protocol.
- 013({X.500_PROTOCOLS}?A:B)When `ds_add_entry(, context,,)` is invoked and **Prefer-Chaining** is set, then chaining is preferred to referrals when necessary. (See GA45 in 7.10.)
Testing Requirements: In a distributed system this assertion can be tested by checking the Directory Access Protocol or the Directory System Protocol.
- 014({X.500_PROTOCOLS}?A:B)When `ds_add_entry(, context,,)` is invoked and **Priority** is set to **low**, then the implementation will attempt to satisfy the request at a low priority relative to the other requests. (See GA46 in 7.10.)
Testing Requirements: In a distributed system this assertion can be tested by checking the Directory Access Protocol or the Directory System Protocol.
- 015({X.500_PROTOCOLS}?A:B)When `ds_add_entry(, context,,)` is invoked and **Priority** is set to **medium**, then the implementation will attempt to satisfy the request at a medium priority relative to the other requests. (See GA47 in 7.10.)
Testing Requirements: In a distributed system this assertion can be tested by checking the Directory Access Protocol or the Directory System Protocol.
- 016({X.500_PROTOCOLS}?A:B)When `ds_add_entry(, context,,)` is invoked and **Priority** is set to **high**, then the implementation will attempt to satisfy the request at a high priority relative to the other requests. (See GA48 in 7.10.)
Testing Requirements: In a distributed system this assertion can be tested by checking the Directory Access Protocol or the Directory System Protocol.
- 017({X.500_PROTOCOLS}?A:B)When `ds_add_entry(, context,,)` is invoked and **Scope-Of-Referral** is set to **country**, then referrals will be limited to DSAs in the originating country. (See GA49 in 7.10.)
Testing Requirements: In a distributed system this assertion can be tested by checking the Directory Access Protocol or the Directory System Protocol.
- 018({X.500_PROTOCOLS}?A:B)When `ds_add_entry(, context,,)` is invoked and **Scope-Of-Referral** is set to **DMD**, then referrals will be limited to DSAs within the originating DMD. (See GA50 in 7.10.)
Testing Requirements: In a distributed system this assertion can be tested by checking the Directory Access Protocol or the Directory System Protocol.
- 019({X.500_PROTOCOLS}?A:B)When `ds_add_entry(, context,,)` is invoked and **Scope-Of-Referral** is not set, then referrals will not be limited. (See GA51 in 7.10.)
Testing Requirements: In a distributed system this assertion can be tested by checking the Directory Access Protocol or the Directory System Protocol.

- 020({X.500_PROTOCOLS}?A:B)When the time specified in **Time-Limit** has elapsed after `ds_add_entry(, context,,)` has been invoked, then the associated directory operation will return a **Service-Error (time-limit-exceeded)**. (See GA53 in 7.10.)
- 021(A) When `ds_add_entry(, context,,)` is invoked and **Asynchronous** is set to **false**, then the associated directory operation will be performed sequentially, the application being blocked until a result or error is returned. (See GA55 in 7.10.)
- 022(C) If the implementation supports asynchronous operation:
When `ds_add_entry(, context,,)` is invoked and **Asynchronous** is set to **true**, then the associated directory operation will be performed asynchronously (nonblocking). (See GA56 in 7.10.)
- 023(B) When `ds_add_entry(, context,,)` is invoked and **Automatic-Continuation** is **false**, then the associated directory operation will return all continuation references to the application. (See GA57 in 7.10.)
- 024(B) When `ds_add_entry(, context,,)` is invoked and **Automatic-Continuation** is **true**, then the associated directory operation will process continuation references automatically and return the subsequent results whenever practical. (See GA58 in 7.10.)

6.3.4 Arguments Out

- 001(C) If the implementation supports asynchronous operation:
When `ds_add_entry` is invoked and the associated directory operation completes successfully and asynchronous operation is selected, then the numerical value returned is guaranteed to be unique among all outstanding directory operations in a given session. The value is such as could be returned from the Remote Operations Service Element (ROSE) defined in ISO/IEC 9072 : 1989 ({3} and {4}). (See GA22 in 5.6.1.)
- 002(C) If the implementation supports asynchronous operation:
When `ds_add_entry` is invoked and `ds_add_entry` completes successfully and asynchronous operation is selected, then the *invoke_id* of the directory operation is returned by `ds_add_entry`. (See GA27 in 5.7.)
- 003(C) If the implementation supports asynchronous operation:
When `ds_add_entry` is invoked and asynchronous operation is selected and completes successfully, then the *invoke_id* returned uniquely identifies the pending result and can be passed to `ds_abandon` to cause the directory operation to be abandoned, or passed to `ds_receive_result` to receive the results for that directory operation. (See GA28 in 5.7.)

6.3.5 Results

- 001(A) When `ds_add_entry` is invoked, then the status result that is returned is one of the following: the constant **success**, the constant **no_workspace**, the constant **invalid_workspace**, an attribute error represented by a private object of class **Attribute-Error**, a referral represented by a private object of class **Referral**, or an error represented by a private object of one of the subclasses of class **Error**. (See GA24 in 5.6.3.)
- 002(A) When `ds_add_entry` is invoked and completes successfully, then the returned value of type `ds_result_type` will be **success**. (See GA92 in 8.1.)
- 003(A) When `ds_add_entry` is invoked and does not complete successfully and has been given a *session* argument containing a valid workspace, then it returns a private object of class **Error**, **Attribute-Error**, or **Referral** in that workspace. (See GA93 in 8.1.)

6.3.6 Errors

- 001(A) When `ds_add_entry` is invoked with a *session* argument that does not contain a valid workspace, then **no_workspace** is returned. (See GA95 in 8.1.)
- 002(C) If the implementation supports asynchronous operation:
When `ds_add_entry` is invoked and asynchronous operation is selected and the number of outstanding directory operations is equal to **Max-Outstanding-Operations**, then it returns an instance of class **Library-Error** with **too-many-operations** value of the **Problem** attribute. (See GA30 in 5.7.)
- 003(D) If automatic connection management is supported:
When `ds_add_entry` is invoked, then the associated directory operation will return the same errors as `ds_bind`. (See GA97 in 8.1.)

- 004(A) When `ds_add_entry` is invoked and an automatic referral list is followed, then the associated directory operation will return the same errors as `ds_bind`. (See GA98 in 8.1.)
- 005(A) When `ds_add_entry` generates an instance of **Attribute-Error**, then it will contain the name of the directory entry to which the directory operation was being applied when the failure occurred in the **Object-Name** attribute. The OM attributes of the instance of **Attribute-Error** that is returned are as stated in R01 in 8.5. (See GA102 in 8.5.)
- 006(A) When `ds_add_entry` generates an instance of **Attribute-Error**, then it will document the attribute-related problems encountered in the **Problems** attribute. The OM attributes of the instance of **Attribute-Error** that is returned are as stated in R01 in 8.5. (See GA103 in 8.5.)
- 007(A) When `ds_add_entry` generates an instance of **Attribute-Error**, then several problems at once can be reported, all related to the **Object-Name** attribute of this class. The OM attributes of the instance of **Attribute-Error** that is returned are as stated in R01 in 8.5. (See GA104 in 8.5.)
- 008(A) When `ds_add_entry` generates an instance of **Attribute-Problem**, then it will identify the type of attribute with which the problem is associated in the **Attribute-Type** attribute. The OM attributes of the instance of **Attribute-Problem** that is returned are as stated in R01 in 8.6. (See GA105 in 8.6.)
- 009(A) When `ds_add_entry` generates an instance of **Attribute-Problem**, then it will contain the attribute value with which the problem is associated, in the **Attribute-Value** attribute. The OM attributes of the instance of **Attribute-Problem** that is returned are as stated in R01 in 8.6. (See GA106 in 8.6.)
- 010(A) When `ds_add_entry` is invoked and an attempt was made to add an attribute or value that is already present in the directory entry in question, then an **Attribute-Problem** error is returned with the value of the **Problem** attribute set to **attribute-or-value-already-exists**. The OM attributes of the instance of **Attribute-Problem** that is returned are as stated in R01 in 8.6. (See GA107 in 8.6.)
- 011(A) When `ds_add_entry` is invoked and the attribute value does not conform to the constraints imposed by the standards (see ISO/IEC 9594-1 : 1990 to ISO/IEC 9594-8 : 1990 {5}-{11}) (e.g., the value exceeds its upper bound), then an **Attribute-Problem** error is returned with the value of the **Problem** attribute set to **constraint-violation**. The OM attributes of the instance of **Attribute-Problem** that is returned are as stated in R01 in 8.6. (See GA108 in 8.6.)
- 012(A) When `ds_add_entry` is invoked and the attribute does not conform to the constraints imposed by the standards (see ISO/IEC 9594-1 : 1990 to ISO/IEC 9594-8 : 1990 {5}-{11}) (e.g., the value exceeds its upper bound), then an **Attribute-Problem** error is returned with the value of the **Problem** attribute set to **constraint-violation**. The OM attributes of the instance of **Attribute-Problem** that is returned are as stated in R01 in 8.6. (See GA109 in 8.6.)
- 013(A) When `ds_add_entry` is invoked and the attribute does not conform to the constraints imposed by the attribute definition (e.g., the value exceeds its upper bound), then an **Attribute-Problem** error is returned with the value of the **Problem** attribute set to **constraint-violation**. The OM attributes of the instance of **Attribute-Problem** that is returned are as stated in R01 in 8.6. (See GA110 in 8.6.)
- 014(A) When `ds_add_entry` is invoked and the attribute value does not conform to the constraints imposed by the attribute definition (e.g., the value exceeds its upper bound), then an **Attribute-Problem** error is returned with the value of the **Problem** attribute set to **constraint-violation**. The OM attributes of the instance of **Attribute-Problem** that is returned are as stated in R01 in 8.6. (See GA111 in 8.6.)
- 015(A) When `ds_add_entry` is invoked and an attempt was made to use a matching rule that is not defined for the attribute type, then an **Attribute-Problem** error is returned with the value of the **Problem** set to **inappropriate-matching**. The OM attributes of the instance of **Attribute-Problem** that is returned are as stated in R01 in 8.6. (See GA112 in 8.6.)
- 016(A) When `ds_add_entry` is invoked and a value presented as an argument does not conform to the attribute syntax of the attribute type, then an **Attribute-Problem** error is returned with the value of the **Problem** attribute set to **invalid-attribute-syntax**. The OM attributes of the instance of **Attribute-Problem** that is returned are as stated in R01 in 8.6. (See GA113 in 8.6.)
- 017(A) When `ds_add_entry` is invoked and the specified attribute or value was not found in the directory entry in question, then an **Attribute-Problem** error is returned with the value of the **Problem** attribute set to **no-such-attribute-or-value**. The OM attributes of the instance of **Attribute-Problem** that is returned are as stated in R01 in 8.6. (See GA114 in 8.6.)
- 018(A) When `ds_add_entry` is invoked and the attribute type that was supplied as an argument is undefined, then an **Attribute-Problem** error is returned with the value of the **Problem** attribute set to **undefined**.

- attribute-type.** The OM attributes of the instance of **Attribute-Problem** that is returned are as stated in R01 in 8.6. (See GA116 in 8.6.)
- 019({X.500_PROTOCOLS}?A:B)When **ds_add_entry** is invoked and a communications error occurs (including those arising in Remote Operation, Association Control, Presentation, Session and Transport), then a **Communications-Error** is returned with the value of the **Problem** attribute set to **communications-problem**. The OM attributes of the instance of **Communications-Error** that is returned are as stated in R01 in 8.7. (See GA117 in 8.7.)
- 020(A) When **ds_add_entry** is invoked and a bad argument (other than name) was supplied, then a **Library-Error** is returned with the value of the **Problem** attribute set to **bad-argument**. The OM attributes of the instance of **Library-Error** that is returned are as stated in R01 in 8.8. (See GA118 in 8.8.)
- 021(A) When **ds_add_entry** is invoked and an instance of OM class **Attribute** was used with no values of the OM attribute **Attribute-Values** as an argument, then a **Library-Error** is returned with the value of the **Problem** attribute set to **bad-argument**. The OM attributes of the instance of **Library-Error** that is returned are as stated in R01 in 8.8. (See GA119 in 8.8.)
- 022(A) When **ds_add_entry** is invoked and the OM class of an argument is not supported for this directory operation, then a **Library-Error** is returned with the value of the **Problem** attribute set to **bad-class**. The OM attributes of the instance of **Library-Error** that is returned are as stated in R01 in 8.8. (See GA120 in 8.8.)
- 023(A) When **ds_add_entry**(, *context*,,) is invoked and an invalid *context* argument was supplied, then a **Library-Error** is returned with the value of the **Problem** attribute set to **bad-context**. The OM attributes of the instance of **Library-Error** that is returned are as stated in R01 in 8.8. (See GA121 in 8.8.)
- 024(A) When **ds_add_entry**(, *name*,,) is invoked and an invalid *name* argument was supplied, then a **Library-Error** is returned with the value of the **Problem** attribute set to **bad-name**. The OM attributes of the instance of **Library-Error** that is returned are as stated in R01 in 8.8. (See GA122 in 8.8.)
- 025(A) When **ds_add_entry**(*session*,,,) is invoked and an invalid *session* was supplied, then a **Library-Error** is returned with the value of the **Problem** attribute set to **bad-session**. The OM attributes of the instance of **Library-Error** that is returned are as stated in R01 in 8.8. (See GA123 in 8.8.)
- 026(A) When **ds_add_entry**(..*session*..) is invoked and is supplied with an argument that has been created in a workspace other than the workspace associated with *session*, then a **Library-Error** is returned with the value of the **Problem** attribute set to **bad-workspace**. The OM attributes of the instance of **Library-Error** that is returned are as stated in R01 in 8.8. (See GA124 in 8.8.)
- 027(B) When **ds_add_entry** is invoked and the interface cannot clear a transient system error by retrying the affected system call, then a **Library-Error** is returned with the value of the **Problem** attribute set to **miscellaneous**. The OM attributes of the instance of **Library-Error** that is returned are as stated in R01 in 8.8. (See GA125 in 8.8.)
- 028(A) When **ds_add_entry**(, *name*,,) is invoked and the attribute type was not included in an attribute-value-assertion passed as part of a distinguished *name* argument, then a **Library-Error** is returned with the value of the **Problem** attribute set to **missing-type**. The OM attributes of the instance of **Library-Error** that is returned are as stated in R01 in 8.8. (See GA126 in 8.8.)
- 029(C) If the implementation supports asynchronous operation:
When **ds_add_entry** is invoked and an attempt to start a synchronous directory operation was made while there were outstanding asynchronous directory operations, then a **Library-Error** is returned with the value of the **Problem** attribute set to **mixed-synchronous**. The OM attributes of the instance of **Library-Error** that is returned are as stated in R01 in 8.8. (See GA127 in 8.8.)
- 030(A) When **ds_add_entry** is invoked and an attempt was made to use optional functionality that is not available in the implementation, then a **Library-Error** is returned with the value of the **Problem** attribute set to **not-supported**. The OM attributes of the instance of **Library-Error** that is returned are as stated in R01 in 8.8. (See GA128 in 8.8.)
- 031(C) If the implementation supports asynchronous operation:
When **ds_add_entry** is invoked and no more directory operations can be performed until at least one asynchronous directory operation has completed, then a **Library-Error** is returned with the value of the **Problem** attribute set to **too-many-operations**. The OM attributes of the instance of **Library-Error** that is returned are as stated in R01 in 8.8. (See GA129 in 8.8.)

- 032(A) When `ds_add_entry(, name,)` generates an instance of **Name-Error**, then the **Matched** attribute will contain the initial portion, up to but excluding the first RDN that is unrecognized, of the *name* that was supplied or of the name resulting from dereferencing an alias. It names the lowest entry (object or alias) in the DIT that was matched. The OM attributes of the instance of **Name-Error** that is returned are as stated in R01 in 8.9. (See GA130 in 8.9.)
- 033(A) When `ds_add_entry` is invoked and an alias was encountered where an alias is not allowed, then a **Name-Error** is returned with the value of the **Problem** attribute set to **alias-dereferencing-problem**. The OM attributes of the instance of **Name-Error** that is returned are as stated in R01 in 8.9. (See GA131 in 8.9.)
- 034(A) When `ds_add_entry` is invoked and an alias that names an object that does not exist has been dereferenced, then a **Name-Error** is returned with the value of the **Problem** attribute set to **alias-problem**. The OM attributes of the instance of **Name-Error** that is returned are as stated in R01 in 8.9. (See GA132 in 8.9.)
- 035(A) When `ds_add_entry` is invoked and the attribute value in an AVA in an RDN in the name does not conform to the attribute syntax prescribed for the attribute type in the AVA, then a **Name-Error** is returned with the value of the **Problem** attribute set to **invalid-attribute-value**. The OM attributes of the instance of **Name-Error** that is returned are as stated in R01 in 8.9. (See GA133 in 8.9.)
- 036(A) When `ds_add_entry(, name,)` is invoked and the specified *name* does not match the name of any object in the directory, then a **Name-Error** is returned with the value of the **Problem** attribute set to **no-such-object**. The OM attributes of the instance of **Name-Error** that is returned are as stated in R01 in 8.9. (See GA134 in 8.9.)
- 037(A) When `ds_add_entry` is invoked and the level of security attached to the credentials of the requester is inconsistent with the level of protection requested, then a **Security-Error** is returned with the value of the **Problem** attribute set to **inappropriate-authentication**. The OM attributes of the instance of **Security-Error** that is returned are as stated in R01 in 8.11. (See GA135 in 8.11.)
- 038(A) When `ds_add_entry` is invoked and the requester does not have permission to perform the directory operation, then a **Security-Error** is returned with the value of the **Problem** attribute set to **insufficient-access-rights**. The OM attributes of the instance of **Security-Error** that is returned are as stated in R01 in 8.11. (See GA136 in 8.11.)
- 039(A) When `ds_add_entry` is invoked and the credentials of the requester are invalid, then a **Security-Error** is returned with the value of the **Problem** attribute set to **invalid-credentials**. The OM attributes of the instance of **Security-Error** that is returned are as stated in R01 in 8.11. (See GA137 in 8.11.)
- 040(A) When `ds_add_entry` is invoked and the signature affixed to the request is invalid, then a **Security-Error** is returned with the value of the **Problem** attribute set to **invalid-signature**. The OM attributes of the instance of **Security-Error** that is returned are as stated in R01 in 8.11. (See GA138 in 8.11.)
- 041(B) When `ds_add_entry` is invoked and the request produced a security error for which no other information is available, then a **Security-Error** is returned with the value of the **Problem** attribute set to **no-information**. The OM attributes of the instance of **Security-Error** that is returned are as stated in R01 in 8.11. (See GA139 in 8.11.)
- 042(A) When `ds_add_entry` is invoked and the directory is unwilling to perform the directory operation because it was unsigned, then a **Security-Error** is returned with the value of the **Problem** attribute set to **protection-required**. The OM attributes of the instance of **Security-Error** that is returned are as stated in R01 in 8.11. (See GA140 in 8.11.)
- 043({X.500_PROTOCOLS}?A:B)When `ds_add_entry` is invoked and the directory operation could not be performed within the administrative constraints on the directory and no partial results are available, then a **Service-Error** is returned with the value of the **Problem** attribute set to **administrative-limit-exceeded**. The OM attributes of the instance of **Service-Error** that is returned are as stated in R01 in 8.12. (See GA141 in 8.12.)
- 044({X.500_PROTOCOLS}?A:B)When `ds_add_entry` is invoked and some part of the directory is temporarily too busy to perform the directory operation, then a **Service-Error** is returned with the value of the **Problem** attribute set to **busy**. The OM attributes of the instance of **Service-Error** that is returned are as stated in R01 in 8.12. (See GA142 in 8.12.)
- 045({X.500_PROTOCOLS}?A:B)When `ds_add_entry` is invoked and chaining is required to perform the directory operation but is prohibited by the **Chaining-Prohibited** service control, then a **Service-Error** is

- returned with the value of the **Problem** attribute set to **chaining-required**. The OM attributes of the instance of **Service-Error** that is returned are as stated in R01 in 8.12. (See GA143 in 8.12.)
- 046({X.500_PROTOCOLS}?A:B)When **ds_add_entry** is invoked and an inconsistency has been detected in the DIT that may be localized to a particular entry or set of entries, then a **Service-Error** is returned with the value of the **Problem** attribute set to **dit-error**. The OM attributes of the instance of **Service-Error** that is returned are as stated in R01 in 8.12. (See GA144 in 8.12.)
- 047({X.500_PROTOCOLS}?A:B)When **ds_add_entry**(, *context*,,) is invoked and the DSA was unable to perform the request as directed (via **Operation-Progress** in the *context*), then a **Service-Error** is returned with the value of the **Problem** attribute set to **invalid-reference**. The OM attributes of the instance of **Service-Error** that is returned are as stated in R01 in 8.12. (See GA145 in 8.12.)
- 048({X.500_PROTOCOLS}?A:B)When **ds_add_entry** is invoked and a DSA detected a loop within the directory, then a **Service-Error** is returned with the value of the **Problem** attribute set to **loop-detected**. The OM attributes of the instance of **Service-Error** that is returned are as stated in R01 in 8.12. (See GA146 in 8.12.)
- 049({X.500_PROTOCOLS}?A:B)When **ds_add_entry** is invoked and the directory cannot provide a referral or partial outcome qualifier within the required scope, then a **Service-Error** is returned with the value of the **Problem** attribute set to **out-of-scope**. The OM attributes of the instance of **Service-Error** that is returned are as stated in R01 in 8.12. (See GA147 in 8.12.)
- 050({X.500_PROTOCOLS}?A:B)When **ds_add_entry** is invoked and the directory operation could not be performed within the time specified by the time-limit service control, and no partial results are available, then a **Service-Error** is returned with the value of the **Problem** attribute set to **time-limit-exceeded**. The OM attributes of the instance of **Service-Error** that is returned are as stated in R01 in 8.12. (See GA148 in 8.12.)
- 051({X.500_PROTOCOLS}?A:B)When **ds_add_entry** is invoked and a DSA without administrative authority over a particular naming context was asked to resolve a name in that context, then a **Service-Error** is returned with the value of the **Problem** attribute set to **unable-to-proceed**. The OM attributes of the instance of **Service-Error** that is returned are as stated in R01 in 8.12. (See GA149 in 8.12.)
- 052({X.500_PROTOCOLS}?A:B)When **ds_add_entry** is invoked and some part of the directory is not currently available, then a **Service-Error** is returned with the value of the **Problem** attribute set to **unavailable**. The OM attributes of the instance of **Service-Error** that is returned are as stated in R01 in 8.12. (See GA150 in 8.12.)
- 053(A) When **ds_add_entry** is invoked and one or more critical extensions were requested but not available, then a **Service-Error** is returned with the value of the **Problem** attribute set to **unavailable-critical-extension**. The OM attributes of the instance of **Service-Error** that is returned are as stated in R01 in 8.12. (See GA151 in 8.12.)
- 054({X.500_PROTOCOLS}?A:B)When **ds_add_entry** is invoked and some part of the directory is not willing to perform the directory operation because it requires excessive resources, then a **Service-Error** is returned with the value of the **Problem** attribute set to **unwilling-to-perform**. The OM attributes of the instance of **Service-Error** that is returned are as stated in R01 in 8.12. (See GA152 in 8.12.)
- 055({X.500_PROTOCOLS}?A:B)When **ds_add_entry** is invoked and some part of the directory is not willing to perform the directory operation because doing so would violate administrative policy, then a **Service-Error** is returned with the value of the **Problem** attribute set to **unwilling-to-perform**. The OM attributes of the instance of **Service-Error** that is returned are as stated in R01 in 8.12. (See GA153 in 8.12.)
- 056(B) When a system error occurs and **ds_add_entry** generates an instance of OM class **System-Error**, then its **Problem** attribute will identify the cause of the failure. The OM attributes of the instance of **System-Error** that is returned are as stated in R01 in 8.13. (See GA154 in 8.13.)
- 057(B) When **ds_add_entry** is invoked and a transient error occurs, then the affected directory operation will be retried and these errors will not be returned. (See GA155 in 8.13.)
- 058({X.500_PROTOCOLS}?A:B)When **ds_add_entry** is invoked and the modification would affect several DSAs (where no local agreement between the DSAs exists), then an **Update-Error** is returned with the value of the **Problem** attribute set to **affects-multiple-DSAs**. The OM attributes of the instance of **Update-Error** that is returned are as stated in R01 in 8.14. (See GA156 in 8.14.)
- 059(A) When **ds_add_entry** is invoked and the modification would leave the DIT improperly structured, then an **Update-Error** is returned with the value of the **Problem** attribute set to **naming-violation**. The OM

attributes of the instance of **Update-Error** that is returned are as stated in R01 in 8.14. (See GA157 in 8.14.)

- 060(A) When `ds_add_entry` is invoked and the modification would be to an interior node of the DIT, then an **Update-Error** is returned with the value of the **Problem** attribute set to **not-allowed-on-non-leaf**. The OM attributes of the instance of **Update-Error** that is returned are as stated in R01 in 8.14. (See GA158 in 8.14.)
- 061(A) When `ds_add_entry` is invoked and the modification would alter the RDN of an object, then an **Update-Error** is returned with the value of the **Problem** attribute set to **not-allowed-on-RDN**. The OM attributes of the instance of **Update-Error** that is returned are as stated in R01 in 8.14. (See GA159 in 8.14.)
- 062(A) When `ds_add_entry` is invoked and the modification would alter the Object Class attribute of an entry, then an **Update-Error** is returned with the value of the **Problem** attribute set to **object-class-modification-prohibited**. The OM attributes of the instance of **Update-Error** that is returned are as stated in R01 in 8.14. (See GA160 in 8.14.)
- 063(A) When `ds_add_entry` is invoked and the modification would leave a directory entry inconsistent with its Object Class definition, then an **Update-Error** is returned with the value of the **Problem** attribute set to **object-class-violation**. The OM attributes of the instance of **Update-Error** that is returned are as stated in R01 in 8.14. (See GA161 in 8.14.)

6.3.7 Cross-References

There are no test assertions for this subclause.

6.4 Bind

6.4.1 Synopsis

- 001(A) `ds_bind` exists in the implementation. (See GA01 in 3.3.2.)
- 002({X.500_PROTOCOLS}?A:B)The Abstract Service **DirectoryBind** defined in the standards maps to `ds_bind`. (See GA03 in 5.1.)

6.4.2 Description

There are no test assertions for this subclause.

6.4.3 Arguments In

- 001(D) If the implementation recognizes a subclass of a class required in an argument (*new_class*) that is an extension:
When `ds_bind(new_class,,)` is invoked, then *new_class* will be recognized. (See GA11 in 5.5.)
- 002(C) If the implementation does not recognize a subclass of a class required in an argument (*new_class*) that is an extension:
When `ds_bind(new_class,,)` is invoked, then it will ignore all OM attributes in *new_class* that are not permitted in the required OM class. (See GA12 in 5.5.)
- 003(A) When `ds_bind(Default-Session,,)` is invoked and completes successfully, then a new session is created with default values for all its OM attributes.
- R01 For `op` in `ds_abandon`, `ds_add_entry`, `ds_bind`, `ds_compare`, `ds_initialize`, `ds_list`, `ds_modify_entry`, `ds_modify_rdn`, `ds_receive_result`, `ds_read`, `ds_remove_entry`, `ds_search`, `ds_unbind`, `ds_version`:
When `op` is invoked using this session and completes successfully, then `op` results will be returned as private objects in this workspace. (See GA21 in 5.6.)
- 004({X.500_PROTOCOLS}?A:B)The Abstract Service **DirectoryBind** defined in the standards has associated arguments that map to the corresponding arguments for `ds_bind`. (See GA09 in 5.5.)

- 005(A) When `ds_bind` is specified to accept an instance of a particular OM class as the value of an argument, then it will also accept an instance of any subclass of the OM class. (See GA10 in 5.5.)

6.4.4 Arguments Out

- 001(A) When `ds_bind` generates an instance of class **Attribute**, then it will contain the directory attribute type in the **Attribute-Type** attribute. The OM attributes of the instance of **Attribute** that is returned are as stated in R01 in 7.5. (See GA37 in 7.5.)
- 002(A) When `ds_bind` generates an instance of class **Attribute**, then it will contain the directory attribute values in the **Attribute-Values** attribute. The OM attributes of the instance of **Attribute** that is returned are as stated in R01 in 7.5. (See GA38 in 7.5.)
- 003(A) When `ds_bind` generates an instance of **DS-DN**, then the **RDNs** attribute contains the sequence of RDNs that define the path through the DIT from its root to the object that the **DS-DN** denotes. The OM attributes of the instance of **DS-DN** that is returned are as stated in R01 in 7.12. (See GA65 in 7.12.)
- 004(A) When `ds_bind` generates an instance of **DS-DN** and the **DS-DN** is representing the root, then the **RDNs** attribute contains the null name (no RDNs values). The OM attributes of the instance of **DS-DN** that is returned are as stated in R01 in 7.12. (See GA66 in 7.12.)
- 005(A) When `ds_bind` generates an instance of **DS-DN**, then the **RDNs** attribute contains the sequence of RDNs of which the first RDN is closest to the root and the last RDN is the value of the object. The OM attributes of the instance of **DS-DN** that is returned are as stated in R01 in 7.12. (See GA67 in 7.12.)
- 006(A) When `ds_bind` generates an instance of **DS-RDN**, then the **AVAs** attribute contains the list of AVAs that are marked by the DIB as components of the RDN of the object. The OM attributes of the instance of **DS-RDN** that is returned are as stated in R01 in 7.13. (See GA68 in 7.13.)
- 007(A) When `ds_bind(, bound_session)` is invoked and completes successfully, then *bound_session* contains an instance of a directory session.
- 008(A) When `ds_bind(session, bound_session)` is invoked and completes successfully and *session* is a public object or takes the value **Default-Session**, then *bound_session* is a new private object.
- 009(A) When `ds_bind(session, bound_session)` is invoked and completes successfully and *session* is not a public object and does not take the value **Default-Session**, then *bound_session* takes the value of that supplied as an argument.
- 010(A) When `ds_bind(session, bound_session)` is invoked and completes successfully, then any OM attributes that were not present in *session* are supplied by `ds_bind(session, bound_session)` as default values in *bound_session*.
- 011(A) When `ds_bind(, bound_session)` is invoked and completes successfully, then `ds_bind(, bound_session)` sets the value of **File-Descriptor** in *bound_session*.
- 012(A) When `ds_bind(, bound_session)` is invoked and completes successfully and the **File-Descriptor** functionality is not supported, then `ds_bind(, bound_session)` sets the value of **File-Descriptor** to **No-Valid-File-Descriptor** in *bound_session*.
- 013(A) When `ds_bind` is invoked and the associated directory operation completes successfully, then `ds_bind` will return private objects in the workspace associated with the directory session in the invocation of `ds_bind`. (See GA21 in 5.6.)

6.4.5 Results

- 001(A) When `ds_bind` is invoked, then the status result that is returned is one of the following: the constant **success**, the constant **no_workspace**, the constant **invalid_workspace**, an attribute error represented by a private object of class **Attribute-Error**, a referral represented by a private object of class **Referral**, or an error represented by a private object of one of the subclasses of class **Error**. (See GA24 in 5.6.3.)
- 002(A) When `ds_bind` is invoked and completes successfully, then the returned value of type `ds_result_type` will be **success**. (See GA92 in 8.1.)
- 003(A) When `ds_bind` is invoked and does not complete successfully and has been given a valid *workspace* argument, then it returns a private object of class **Error**, **Attribute-Error**, or **Referral** in that workspace. (See GA94 in 8.1.)

6.4.6 Errors

- 001(A) When `ds_bind` is invoked with a *workspace* argument that is invalid, then **invalid_workspace** is returned. (See GA96 in 8.1.)
- 002({X.500_PROTOCOLS}?A:B)When `ds_bind` is invoked and a communications error occurs (including those arising in Remote Operation, Association Control, Presentation, Session, and Transport), then a **Communications-Error** is returned with the value of the **Problem** attribute set to **communications-problem**. The OM attributes of the instance of **Communications-Error** that is returned are as stated in R01 in 8.7. (See GA117 in 8.7.)
- 003(A) When `ds_bind` is invoked and the OM class of an argument is not supported for this directory operation, then a **Library-Error** is returned with the value of the **Problem** attribute set to **bad-class**. The OM attributes of the instance of **Library-Error** that is returned are as stated in R01 in 8.8. (See GA120 in 8.8.)
- 004(A) When `ds_bind(session,,)` is invoked and an invalid *session* was supplied, then a **Library-Error** is returned with the value of the **Problem** attribute set to **bad-session**. The OM attributes of the instance of **Library-Error** that is returned are as stated in R01 in 8.8. (See GA123 in 8.8.)
- 005(B) When `ds_bind` is invoked and the interface cannot clear a transient system error by retrying the affected system call, then a **Library-Error** is returned with the value of the **Problem** attribute set to **miscellaneous**. The OM attributes of the instance of **Library-Error** that is returned are as stated in R01 in 8.8. (See GA125 in 8.8.)
- 006(C) If the implementation supports asynchronous operation:
When `ds_bind` is invoked and an attempt to start a synchronous directory operation was made while there were outstanding asynchronous directory operations, then a **Library-Error** is returned with the value of the **Problem** attribute set to **mixed-synchronous**. The OM attributes of the instance of **Library-Error** that is returned are as stated in R01 in 8.8. (See GA127 in 8.8.)
- 007(A) When `ds_bind` is invoked and an attempt was made to use optional functionality that is not available in the implementation, then a **Library-Error** is returned with the value of the **Problem** attribute set to **not-supported**. The OM attributes of the instance of **Library-Error** that is returned are as stated in R01 in 8.8. (See GA128 in 8.8.)
- 008(A) When `ds_bind` is invoked and the level of security attached to the credentials of the requester is inconsistent with the level of protection requested, then **Security-Error** is returned with the value of the **Problem** attribute set to **inappropriate-authentication**. The OM attributes of the instance of **Security-Error** that is returned are as stated in R01 in 8.11. (See GA135 in 8.11.)
- 009(A) When `ds_bind` is invoked and the requester does not have permission to perform the directory operation, then a **Security-Error** is returned with the value of the **Problem** attribute set to **insufficient-access-rights**. The OM attributes of the instance of **Security-Error** that is returned are as stated in R01 in 8.11. (See GA136 in 8.11.)
- 010(A) When `ds_bind` is invoked and the credentials of the requester are invalid, then a **Security-Error** is returned with the value of the **Problem** attribute set to **invalid-credentials**. The OM attributes of the instance of **Security-Error** that is returned are as stated in R01 in 8.11. (See GA137 in 8.11.)
- 011(A) When `ds_bind` is invoked and the signature affixed to the request is invalid, then a **Security-Error** is returned with the value of the **Problem** attribute set to **invalid-signature**. The OM attributes of the instance of **Security-Error** that is returned are as stated in R01 in 8.11. (See GA138 in 8.11.)
- 012(B) When `ds_bind` is invoked and the request produced a security error for which no other information is available, then a **Security-Error** is returned with the value of the **Problem** attribute set to **no-information**. The OM attributes of the instance of **Security-Error** that is returned are as stated in R01 in 8.11. (See GA139 in 8.11.)
- 013(A) When `ds_bind` is invoked and the directory is unwilling to perform the directory operation because it was unsigned, then a **Security-Error** is returned with the value of the **Problem** attribute set to **protection-required**. The OM attributes of the instance of **Security-Error** that is returned are as stated in R01 in 8.11. (See GA140 in 8.11.)
- 014({X.500_PROTOCOLS}?A:B)When `ds_bind` is invoked and the directory operation could not be performed within the administrative constraints on the directory and no partial results are available, then a **Service-Error** is returned with the value of the **Problem** attribute set to **administrative-limit-exceeded**. The OM

- attributes of the instance of **Service-Error** that is returned are as stated in R01 in 8.12. (See GA141 in 8.12.)
- 015({X.500_PROTOCOLS}?A:B)When **ds_bind** is invoked and some part of the directory is temporarily too **busy** to perform the directory operation, then a **Service-Error** is returned with the value of the **Problem** attribute set to **busy**. The OM attributes of the instance of **Service-Error** that is returned are as stated in R01 in 8.12. (See GA142 in 8.12.)
- 016({X.500_PROTOCOLS}?A:B)When **ds_bind** is invoked and an inconsistency has been detected in the DIT that may be localized to a particular entry or set of entries, then a **Service-Error** is returned with the value of the **Problem** attribute set to **dit-error**. The OM attributes of the instance of **Service-Error** that is returned are as stated in R01 in 8.12. (See GA144 in 8.12.)
- 017({X.500_PROTOCOLS}?A:B)When **ds_bind** is invoked and a DSA detected a loop within the directory, then a **Service-Error** is returned with the value of the **Problem** attribute set to **loop-detected**. The OM attributes of the instance of **Service-Error** that is returned are as stated in R01 in 8.12. (See GA146 in 8.12.)
- 018({X.500_PROTOCOLS}?A:B)When **ds_bind** is invoked and the directory cannot provide a referral or partial outcome qualifier within the required scope, then a **Service-Error** is returned with the value of the **Problem** attribute set to **out-of-scope**. The OM attributes of the instance of **Service-Error** that is returned are as stated in R01 in 8.12. (See GA147 in 8.12.)
- 019({X.500_PROTOCOLS}?A:B)When **ds_bind** is invoked and a DSA without administrative authority over a particular naming context was asked to resolve a name in that context, then a **Service-Error** is returned with the value of the **Problem** attribute set to **unable-to-proceed**. The OM attributes of the instance of **Service-Error** that is returned are as stated in R01 in 8.12. (See GA149 in 8.12.)
- 020({X.500_PROTOCOLS}?A:B)When **ds_bind** is invoked and some part of the directory is not currently available, then a **Service-Error** is returned with the value of the **Problem** attribute set to **unavailable**. The OM attributes of the instance of **Service-Error** that is returned are as stated in R01 in 8.12. (See GA150 in 8.12.)
- 021({X.500_PROTOCOLS}?A:B)When **ds_bind** is invoked and some part of the directory is not willing to perform the directory operation because it requires excessive resources, then a **Service-Error** is returned with the value of the **Problem** attribute set to **unwilling-to-perform**. The OM attributes of the instance of **Service-Error** that is returned are as stated in R01 in 8.12. (See GA152 in 8.12.)
- 022({X.500_PROTOCOLS}?A:B)When **ds_bind** is invoked and some part of the directory is not willing to perform the directory operation because doing so would violate administrative policy, then a **Service-Error** is returned with the value of the **Problem** attribute set to **unwilling-to-perform**. The OM attributes of the instance of **Service-Error** that is returned are as stated in R01 in 8.12. (See GA153 in 8.12.)
- 023(B) When a system error occurs and **ds_bind** generates an instance of OM class **System-Error**, then its **Problem** attribute will identify the cause of the failure. The OM attributes of the instance of **System-Error** that is returned are as stated in R01 in 8.13. (See GA154 in 8.13.)
- 024(B) When **ds_bind** is invoked and a transient error occurs, then the affected directory operation will be retried and these errors will not be returned. (See GA155 in 8.13.)

6.4.7 Cross-References

There are no test assertions for this subclause.

6.5 Compare

6.5.1 Synopsis

- 001(A) **ds_compare** exists in the implementation. (See GA01 in 3.3.2.)

6.5.2 Description

- 001(A) When `ds_compare(, name,ava,,)` is invoked and the associated directory operation completes successfully, then the value supplied in the parameter `ava` is compared with the values of the same attribute type in the parameter `name`.
- 002({X.500_PROTOCOLS}?A:B)The Abstract Service Compare defined in the standards maps to `ds_compare`. (See GA02 in 5.1.)
- 003(C) If the implementation does not support asynchronous operation:
The directory operation associated with `ds_compare` is synchronous. (See GA25 in 5.7.)
- 004(C) If the implementation supports asynchronous operation:
When `ds_compare` is invoked and asynchronous operation is selected and an error is detected before the associated asynchronous request is submitted to the directory, then `ds_compare` returns control immediately and no outstanding directory operation is generated. (See GA31 in 5.7.)
- 005(C) If the implementation supports asynchronous operation:
When `ds_compare` is invoked and asynchronous operation is selected and an error occurs after the associated asynchronous request is submitted to the directory, then errors are reported by `ds_receive_result`. (See GA32 in 5.7.)
- 006(A) When continuation references have all been used by `ds_compare` to attempt to contact other DSAs, then `ds_compare` returns a result or error. (See GA33 in 5.9.)

6.5.3 Arguments In

- 001(D) If the implementation recognizes a subclass of a class required in an argument (*new_class*) that is an extension:
When `ds_compare(.. new_class..)` is invoked, then *new_class* will be recognized. (See GA11 in 5.5.)
- 002(C) If the implementation does not recognize a subclass of a class required in an argument (*new_class*) that is an extension:
When `ds_compare(.. new_class..)` is invoked, then it will ignore all OM attributes in *new_class* that are not permitted in the required OM class. (See GA12 in 5.5.)
- 003(B) When `ds_compare` is invoked, then the **ServiceControls** defined in the Abstract Service each map onto an OM attribute in the **Context** that is used. (See GA05 in 5.4.)
- 004(A) When `ds_compare` is invoked, then the value of each component of the **Context** is determined and remains fixed throughout the directory operation. (See GA06 in 5.4.)
- 005(A) When `ds_compare` is invoked, passing **Default-Context** as the *Context* argument, then it has the same effect as passing the **Context** object with default values. (See GA07 in 5.4.)
- 006({X.500_PROTOCOLS}?A:B)The Abstract Service Compare defined in the standards has associated arguments that map to the corresponding arguments for `ds_compare`. (See GA08 in 5.5.)
- 007(A) When `ds_compare` is specified to accept an instance of a particular OM class as the value of an argument, then it will also accept an instance of any subclass of the OM class. (See GA10 in 5.5.)
- 008(C) If the implementation supports asynchronous operation:
When the value of the **Asynchronous** OM attribute is not specified in the *Context* passed to `ds_compare`, then `ds_compare` behaves as if the value of the OM attribute was **false** and the directory operation is synchronous. (See GA26 in 5.7.)
- 009(D) If the implementation defines that all interface operations return control after an asynchronous directory operation has been submitted to the underlying directory service:
When `ds_compare` is invoked and asynchronous operation is selected, then it returns control after waiting for the associated directory operation to be submitted to the underlying directory service.
Otherwise:
When `ds_compare` is invoked and asynchronous operation is selected, then it returns control as soon as possible. (See GA29 in 5.7.)
- 010({X.500_PROTOCOLS}?A:B)When `ds_compare(, context,,,)` is invoked and **Chaining-Prohibited** is set, then chaining and other methods of distributing the request around the directory are prohibited. (See GA41 in 7.10.)

Testing Requirements: In a distributed system this assertion can be tested by checking the Directory Access Protocol or the Directory System Protocol.

011(A) When `ds_compare(, context,,)` is invoked and **Dont-Dereference-Aliases** is set, then any alias used to identify the target entry of a directory operation will not be dereferenced. (See GA42 in 7.10.)

012({X.500_PROTOCOLS}?A:B)When `ds_compare(, context,,)` is invoked and **Dont-Use-Copy** is set, then the request is to be satisfied only by access to directory entries and not copies maintained in other DSAs or locally cached copies. (See GA43 in 7.10.)

Testing Requirements: In a distributed system this assertion can be tested by checking the Directory Access Protocol or the Directory System Protocol.

013({X.500_PROTOCOLS}?A:B)When `ds_compare(, context,,)` is invoked and **Local-Scope** is set, then the request is to be satisfied locally, the meaning of which is configured by an administrator. (See GA44 in 7.10.)

Testing Requirements: In a distributed system this assertion can be tested by checking the Directory Access Protocol or the Directory System Protocol.

014({X.500_PROTOCOLS}?A:B)When `ds_compare(,context,,)` is invoked and **Prefer-Chaining** is set, then chaining is preferred to referrals when necessary. (See GA45 in 7.10.)

Testing Requirements: In a distributed system this assertion can be tested by checking the Directory Access Protocol or the Directory System Protocol.

015({X.500_PROTOCOLS}?A:B)When `ds_compare(, context,,)` is invoked and **Priority** is set to **low**, then the implementation will attempt to satisfy the request at a low priority relative to the other requests. (See GA46 in 7.10.)

Testing Requirements: In a distributed system this assertion can be tested by checking the Directory Access Protocol or the Directory System Protocol.

016({X.500_PROTOCOLS}?A:B)When `ds_compare(, context,,)` is invoked and **Priority** is set to **medium**, then the implementation will attempt to satisfy the request at a medium priority relative to the other requests. (See GA47 in 7.10.)

Testing Requirements: In a distributed system this assertion can be tested by checking the Directory Access Protocol or the Directory System Protocol.

017({X.500_PROTOCOLS}?A:B)When `ds_compare(, context,,)` is invoked and **Priority** is set to **high**, then the implementation will attempt to satisfy the request at a high priority relative to the other requests. (See GA48 in 7.10.)

Testing Requirements: In a distributed system this assertion can be tested by checking the Directory Access Protocol or the Directory System Protocol.

018({X.500_PROTOCOLS}?A:B)When `ds_compare(, context,,)` is invoked and **Scope-Of-Referral** is set to **country**, then referrals will be limited to DSAs in the originating country. (See GA49 in 7.10.)

Testing Requirements: In a distributed system this assertion can be tested by checking the Directory Access Protocol or the Directory System Protocol.

019({X.500_PROTOCOLS}?A:B)When `ds_compare(, context,,)` is invoked and **Scope-Of-Referral** is set to **DMD**, then referrals will be limited to DSAs within the originating DMD. (See GA50 in 7.10.)

Testing Requirements: In a distributed system this assertion can be tested by checking the Directory Access Protocol or the Directory System Protocol.

020({X.500_PROTOCOLS}?A:B)When `ds_compare(, context,,)` is invoked and **Scope-Of-Referral** is not set, then referrals will not be limited. (See GA51 in 7.10.)

Testing Requirements: In a distributed system this assertion can be tested by checking the Directory Access Protocol or the Directory System Protocol.

021({X.500_PROTOCOLS}?A:B)When the time specified in **Time-Limit** has elapsed after `ds_compare(, context,,)` has been invoked, then the associated directory operation will return a **Service-Error (time-limit-exceeded)**. (See GA53 in 7.10.)

022(A) When `ds_compare(, context,,)` is invoked and **Asynchronous** is set to **false**, then the associated directory operation will be performed sequentially, the application being blocked until a result or error is returned. (See GA55 in 7.10.)

023(C) If the implementation supports asynchronous operation:

When `ds_compare(, context,,)` is invoked and **Asynchronous** is set to **true**, then the associated directory operation will be performed asynchronously (nonblocking). (See GA56 in 7.10.)

- 024(B) When `ds_compare(, context,,)` is invoked and **Automatic-Continuation** is **false**, then the associated directory operation will return all continuation references to the application. (See GA57 in 7.10.)
- 025(B) When `ds_compare(, context,,)` is invoked and **Automatic-Continuation** is **true**, then the associated directory operation will process continuation references automatically and return the subsequent results whenever practical. (See GA58 in 7.10.)

6.5.4 Arguments Out

- R01 When `ds_compare(,,, result,)` is invoked and the associated directory operation completed successfully, then *result* contains flags indicating whether the values matched. The OM attributes of the instance of **Compare-Result** that is returned are as stated in R01 in 7.9. [See 003(A) in 7.9 and 004(A) in 7.9.]
- R02 When `ds_compare(,,, result)` is invoked and the associated directory operation completed successfully, then *result* contains flags indicating whether the comparison was made against the original entry. The OM attributes of the instance of **Compare-Result** that is returned are as stated in R01 in 7.9. [See 002(A) in 7.9.]
- R03 When `ds_compare(,,, result,)` is invoked and the associated directory operation completed successfully and an alias was dereferenced, then *result* contains the distinguished name of the target object. The OM attributes of the instance of **Compare-Result** that is returned are as stated in R01 in 7.9. [See 005(A) in 7.9.]
- 001(A) When the workspace returned by `ds_initialize` (and not subsequently extended) is used by `ds_compare`, then any directory attribute values returned by `ds_compare` will be represented by OM objects of class **Encoding**. (See GA04 in 5.2.)
- 002(C) If the implementation supports the Basic Directory Contents Package:
When the use of the BDCP has been negotiated and `ds_compare` is invoked and completes successfully, returning any attribute values that are defined in the BDCP, then they are represented as specified in the BDCP. (See GA13 in 5.5.1.)
- 003(C) If the implementation supports the Strong Authentication Package:
When the use of the SAP has been negotiated and `ds_compare` is invoked and completes successfully, returning any attribute values that are defined in the SAP, then they are represented as specified in the SAP. (See GA14 in 5.5.1.)
- 004(A) When `ds_compare` is invoked and completes successfully, returning any attribute values that are of ASN.1 simple type and are not known, then they are represented as shown in the table below. (See GA15 in 5.5.1.)

Type	Syntax
Bit String	String(Bit)
Boolean	Boolean
Integer	Integer
Null	Null
Object Identifier	String(Object Identifier)

- 005(A) When `ds_compare` is invoked and completes successfully, returning any attribute values whose attribute type is not known that is an ASN.1 structured type, then they are represented by OM objects of class **Encoding**. (See GA16 in 5.5.1.)
- 006(A) When `ds_compare` is invoked and the associated directory operation completes successfully, then `ds_compare` will return private objects in the workspace associated with the directory session in the invocation of `ds_bind`. (See GA21 in 5.6.)
- 007(C) If the implementation supports asynchronous operation:
When `ds_compare` is invoked and the associated directory operation completes successfully and asynchronous operation is selected, then the numerical value returned is guaranteed to be unique among all

- outstanding directory operations in a given session. The value is such as could be returned from the Remote Operations Service Element (ROSE) defined in ISO/IEC 9072 : 1989 ({3} and {4}). (See GA22 in 5.6.1.)
- 008(A) When `ds_compare` is invoked and the associated directory operation does not complete successfully, then a result is not returned. (See GA23 in 5.6.2.)
- 009(C) If the implementation supports asynchronous operation:
When `ds_compare` is invoked and `ds_compare` completes successfully and asynchronous operation is selected, then the *invoke_id* of the directory operation is returned by `ds_compare`. (See GA27 in 5.7.)
- 010(C) If the implementation supports asynchronous operation:
When `ds_compare` is invoked and asynchronous operation is selected and completes successfully, then the *invoke_id* returned uniquely identifies the pending result and can be passed to `ds_abandon` to cause the directory operation to be abandoned, or passed to `ds_receive_result` to receive the results for that directory operation. (See GA28 in 5.7.)
- 011(A) When `ds_compare` generates an instance of class **Attribute**, then it will contain the directory attribute type in the **Attribute-Type** attribute. The OM attributes of the instance of **Attribute** that is returned are as stated in R01 in 7.5. (See GA37 in 7.5.)
- 012(A) When `ds_compare` generates an instance of class **Attribute**, then it will contain the directory attribute values in the **Attribute-Values** attribute. The OM attributes of the instance of **Attribute** that is returned are as stated in R01 in 7.5. (See GA38 in 7.5.)
- 013(A) When `ds_compare` generates an instance of class **AVA**, then the **Attribute-Values** attribute of its superclass **Attribute** will be constrained to have only one value. The OM attributes of the instance of **AVA** that is returned are as stated in R01 in 7.7. (See GA40 in 7.7.)
- 014(A) When `ds_compare` generates an instance of **DS-DN**, then the **RDNs** attribute contains the sequence of RDNs that define the path through the DIT from its root to the object that the **DS-DN** denotes. The OM attributes of the instance of **DS-DN** that is returned are as stated in R01 in 7.12. (See GA65 in 7.12.)
- 015(A) When `ds_compare` generates an instance of **DS-DN** and the **DS-DN** is representing the root, then the **RDNs** attribute contains the null name (no RDNs values). The OM attributes of the instance of **DS-DN** that is returned are as stated in R01 in 7.12. (See GA66 in 7.12.)
- 016(A) When `ds_compare` generates an instance of **DS-DN**, then the **RDNs** attribute contains the sequence of RDNs of which the first RDN is closest to the root and the last RDN is the value of the object. The OM attributes of the instance of **DS-DN** that is returned are as stated in R01 in 7.12. (See GA67 in 7.12.)
- 017(A) When `ds_compare` generates an instance of **DS-RDN**, then the **AVAs** attribute contains the list of AVAs that are marked by the DIB as components of the RDN of the object. The OM attributes of the instance of **DS-RDN** that is returned are as stated in R01 in 7.13. (See GA68 in 7.13.)

6.5.5 Results

- 001(A) When `ds_compare` is invoked, then the status result that is returned is one of the following: the constant **success**, the constant **no_workspace**, the constant **invalid_workspace**, an attribute error represented by a private object of class **Attribute-Error**, a referral represented by a private object of class **Referral**, or an error represented by a private object of one of the subclasses of class **Error**. (See GA24 in 5.6.3.)
- 002(A) When `ds_compare` is invoked and completes successfully, then the returned value of type `ds_result_type` will be **success**. (See GA92 in 8.1.)
- 003(A) When `ds_compare` is invoked and does not complete successfully and has been given a *session* argument containing a valid workspace, then it returns a private object of class **Error**, **Attribute-Error**, or **Referral** in that workspace. (See GA93 in 8.1.)

6.5.6 Errors

- 001(C) If the implementation supports asynchronous operation:
When `ds_compare` is invoked and asynchronous operation is selected and the number of outstanding directory operations is equal to **Max-Outstanding-Operations**, then it returns an instance of class **Library-Error** with **too-many-operations** value of the **Problem** attribute. (See GA30 in 5.7.)
- 002(A) When `ds_compare` is invoked with a *session* argument that does not contain a valid workspace, then **no_workspace** is returned. (See GA95 in 8.1.)

- 003(D) If automatic connection management is supported:
When `ds_compare` is invoked, then the associated directory operation will return the same errors as `ds_bind`. (See GA97 in 8.1.)
- 004(A) When `ds_compare` is invoked and an automatic referral list is followed, then the associated directory operation will return the same errors as `ds_bind`. (See GA98 in 8.1.)
- 005(A) When `ds_compare` generates an instance of **Attribute-Error**, then it will contain the name of the directory entry to which the directory operation was being applied when the failure occurred in the **Object-Name** attribute. The OM attributes of the instance of **Attribute-Error** that is returned are as stated in R01 in 8.5. (See GA102 in 8.5.)
- 006(A) When `ds_compare` generates an instance of **Attribute-Error**, then it will document the attribute-related problems encountered in the **Problems** attribute. The OM attributes of the instance of **Attribute-Error** that is returned are as stated in R01 in 8.5. (See GA103 in 8.5.)
- 007(A) When `ds_compare` generates an instance of **Attribute-Error**, then several problems at once can be reported, all related to the **Object-Name** attribute of this class. The OM attributes of the instance of **Attribute-Error** that is returned are as stated in R01 in 8.5. (See GA104 in 8.5.)
- 008(A) When `ds_compare` generates an instance of **Attribute-Problem**, then it will identify the type of attribute with which the problem is associated in the **Attribute-Type** attribute. The OM attributes of the instance of **Attribute-Problem** that is returned are as stated in R01 in 8.6. (See GA105 in 8.6.)
- 009(A) When `ds_compare` generates an instance of **Attribute-Problem**, then it will contain the attribute value with which the problem is associated in the **Attribute-Value** attribute. The OM attributes of the instance of **Attribute-Problem** that is returned are as stated in R01 in 8.6. (See GA106 in 8.6.)
- 010(A) When `ds_compare` is invoked and an attempt was made to add an attribute or value that is already present in the directory entry in question, then an **Attribute-Problem** error is returned with the value of the **Problem** attribute set to **attribute-or-value-already-exists**. The OM attributes of the instance of **Attribute-Problem** that is returned are as stated in R01 in 8.6. (See GA107 in 8.6.)
- 011(A) When `ds_compare` is invoked and the attribute value does not conform to the constraints imposed by the standards (see ISO/IEC 9594-1 : 1990 to ISO/IEC 9594-8 : 1990 {5}–{11}) (e.g., the value exceeds its upper bound), then an **Attribute-Problem** error is returned with the value of the **Problem** attribute set to **constraint-violation**. The OM attributes of the instance of **Attribute-Problem** that is returned are as stated in R01 in 8.6. (See GA108 in 8.6.)
- 012(A) When `ds_compare` is invoked and the attribute does not conform to the constraints imposed by the standards (see ISO/IEC 9594-1 : 1990 to ISO/IEC 9594-8 : 1990 {5}–{11}) (e.g., the value exceeds its upper bound), then an **Attribute-Problem** error is returned with the value of the **Problem** attribute set to **constraint-violation**. The OM attributes of the instance of **Attribute-Problem** that is returned are as stated in R01 in 8.6. (See GA109 in 8.6.)
- 013(A) When `ds_compare` is invoked and the attribute does not conform to the constraints imposed by the attribute definition (e.g., the value exceeds its upper bound), then an **Attribute-Problem** error is returned with the value of the **Problem** attribute set to **constraint-violation**. The OM attributes of the instance of **Attribute-Problem** that is returned are as stated in R01 in 8.6. (See GA110 in 8.6.)
- 014(A) When `ds_compare` is invoked and the attribute value does not conform to the constraints imposed by the attribute definition (e.g., the value exceeds its upper bound), then an **Attribute-Problem** error is returned with the value of the **Problem** attribute set to **constraint-violation**. The OM attributes of the instance of **Attribute-Problem** that is returned are as stated in R01 in 8.6. (See GA111 in 8.6.)
- 015(A) When `ds_compare` is invoked and an attempt was made to use a matching rule that is not defined for the attribute type, then an **Attribute-Problem** error is returned with the value of the **Problem** set to **inappropriate-matching**. The OM attributes of the instance of **Attribute-Problem** that is returned are as stated in R01 in 8.6. (See GA112 in 8.6.)
- 016(A) When `ds_compare` is invoked and a value presented as an argument does not conform to the attribute syntax of the attribute type, then an **Attribute-Problem** error is returned with the value of the **Problem** attribute set to **invalid-attribute-syntax**. The OM attributes of the instance of **Attribute-Problem** that is returned are as stated in R01 in 8.6. (See GA113 in 8.6.)
- 017(A) When `ds_compare` is invoked and the specified attribute or value was not found in the directory entry in question, then an **Attribute-Problem** error is returned with the value of the **Problem** attribute set to

- no-such-attribute-or-value.** The OM attributes of the instance of **Attribute-Problem** that is returned are as stated in R01 in 8.6. (See GA114 in 8.6.)
- 018({X.500_PROTOCOLS}?A:B)When **ds_compare** is invoked and a communications error occurs (including those arising in Remote Operation, Association Control, Presentation, Session and Transport), then a **Communications-Error** is returned with the value of the **Problem** attribute set to **communications-problem**. The OM attributes of the instance of **Communications-Error** that is returned are as stated in R01 in 8.7. (See GA117 in 8.7.)
- 019(A) When **ds_compare** is invoked and a bad argument (other than name) was supplied, then a **Library-Error** is returned with the value of the **Problem** attribute set to **bad-argument**. The OM attributes of the instance of **Library-Error** that is returned are as stated in R01 in 8.8. (See GA118 in 8.8.)
- 020(A) When **ds_compare** is invoked and an instance of OM class **Attribute** was used with no values of the OM attribute **Attribute-Values** as an argument, then a **Library-Error** is returned with the value of the **Problem** attribute set to **bad-argument**. The OM attributes of the instance of **Library-Error** that is returned are as stated in R01 in 8.8. (See GA119 in 8.8.)
- 021(A) When **ds_compare** is invoked and the OM class of an argument is not supported for this directory operation, then a **Library-Error** is returned with the value of the **Problem** attribute set to **bad-class**. The OM attributes of the instance of **Library-Error** that is returned are as stated in R01 in 8.8. (See GA120 in 8.8.)
- 022(A) When **ds_compare**(, *context*,,,) is invoked and an invalid *context* argument was supplied, then a **Library-Error** is returned with the value of the **Problem** attribute set to **bad-context**. The OM attributes of the instance of **Library-Error** that is returned are as stated in R01 in 8.8. (See GA121 in 8.8.)
- 023(A) When **ds_compare**(, *name*,,,) is invoked and an invalid *name* argument was supplied, then a **Library-Error** is returned with the value of the **Problem** attribute set to **bad-name**. The OM attributes of the instance of **Library-Error** that is returned are as stated in R01 in 8.8. (See GA122 in 8.8.)
- 024(A) When **ds_compare**(*session*,,,,) is invoked and an invalid *session* was supplied, then a **Library-Error** is returned with the value of the **Problem** attribute set to **bad-session**. The OM attributes of the instance of **Library-Error** that is returned are as stated in R01 in 8.8. (See GA123 in 8.8.)
- 025(A) When **ds_compare**(..*session*..) is invoked and is supplied with an argument that has been created in a workspace other than the workspace associated with *session*, then a **Library-Error** is returned with the value of the **Problem** attribute set to **bad-workspace**. The OM attributes of the instance of **Library-Error** that is returned are as stated in R01 in 8.8. (See GA124 in 8.8.)
- 026(B) When **ds_compare** is invoked and the interface cannot clear a transient system error by retrying the affected system call, then a **Library-Error** is returned with the value of the **Problem** attribute set to **miscellaneous**. The OM attributes of the instance of **Library-Error** that is returned are as stated in R01 in 8.8. (See GA125 in 8.8.)
- 027(A) When **ds_compare**(, *name*,,,) is invoked and the attribute type was not included in an attribute-value-assertion passed as part of a distinguished *name* argument, then a **Library-Error** is returned with the value of the **Problem** attribute set to **missing-type**. The OM attributes of the instance of **Library-Error** that is returned are as stated in R01 in 8.8. (See GA126 in 8.8.)
- 028(C) If the implementation supports asynchronous operation:
When **ds_compare** is invoked and an attempt to start a synchronous directory operation was made while there were outstanding asynchronous directory operations, then a **Library-Error** is returned with the value of the **Problem** attribute set to **mixed-synchronous**. The OM attributes of the instance of **Library-Error** that is returned are as stated in R01 in 8.8. (See GA127 in 8.8.)
- 029(A) When **ds_compare** is invoked and an attempt was made to use optional functionality that is not available in the implementation, then a **Library-Error** is returned with the value of the **Problem** attribute set to **not-supported**. The OM attributes of the instance of **Library-Error** that is returned are as stated in R01 in 8.8. (See GA128 in 8.8.)
- 030(C) If the implementation supports asynchronous operation:
When **ds_compare** is invoked and no more directory operations can be performed until at least one asynchronous directory operation has completed, then a **Library-Error** is returned with the value of the **Problem** attribute set to **too-many-operations**. The OM attributes of the instance of **Library-Error** that is returned are as stated in R01 in 8.8. (See GA129 in 8.8.)

- 031(A) When `ds_compare(, name,,)` generates an instance of **Name-Error**, then the **Matched** attribute will contain the initial portion, up to but excluding the first RDN that is unrecognized, of the *name* that was supplied or of the name resulting from dereferencing an alias. It names the lowest entry (object or alias) in the DIT that was matched. The OM attributes of the instance of **Name-Error** that is returned are as stated in R01 in 8.9. (See GA130 in 8.9.)
- 032(A) When `ds_compare` is invoked and an alias was encountered where an alias is not allowed, then a **Name-Error** is returned with the value of the **Problem** attribute set to **alias-dereferencing-problem**. The OM attributes of the instance of **Name-Error** that is returned are as stated in R01 in 8.9. (See GA131 in 8.9.)
- 033(A) When `ds_compare` is invoked and an alias that names an object that does not exist has been dereferenced, then a **Name-Error** is returned with the value of the **Problem** attribute set to **alias-problem**. The OM attributes of the instance of **Name-Error** that is returned are as stated in R01 in 8.9. (See GA132 in 8.9.)
- 034(A) When `ds_compare` is invoked and the attribute value in an AVA in an RDN in the name does not conform to the attribute syntax prescribed for the attribute type in the AVA, then a **Name-Error** is returned with the value of the **Problem** attribute set to **invalid-attribute-value**. The OM attributes of the instance of **Name-Error** that is returned are as stated in R01 in 8.9. (See GA133 in 8.9.)
- 035(A) When `ds_compare(, name,,)` is invoked and the specified *name* does not match the name of any object in the directory, then a **Name-Error** is returned with the value of the **Problem** attribute set to **no-such-object**. The OM attributes of the instance of **Name-Error** that is returned are as stated in R01 in 8.9. (See GA134 in 8.9.)
- 036(A) When `ds_compare` is invoked and the level of security attached to the credentials of the requester is inconsistent with the level of protection requested, then a **Security-Error** is returned with the value of the **Problem** attribute set to **inappropriate-authentication**. The OM attributes of the instance of **Security-Error** that is returned are as stated in R01 in 8.11. (See GA135 in 8.11.)
- 037(A) When `ds_compare` is invoked and the requester does not have permission to perform the directory operation, then a **Security-Error** is returned with the value of the **Problem** attribute set to **insufficient-access-rights**. The OM attributes of the instance of **Security-Error** that is returned are as stated in R01 in 8.11. (See GA136 in 8.11.)
- 038(A) When `ds_compare` is invoked and the credentials of the requester are invalid, then a **Security-Error** is returned with the value of the **Problem** attribute set to **invalid-credentials**. The OM attributes of the instance of **Security-Error** that is returned are as stated in R01 in 8.11. (See GA137 in 8.11.)
- 039(A) When `ds_compare` is invoked and the signature affixed to the request is invalid, then a **Security-Error** is returned with the value of the **Problem** attribute set to **invalid-signature**. The OM attributes of the instance of **Security-Error** that is returned are as stated in R01 in 8.11. (See GA138 in 8.11.)
- 040(B) When `ds_compare` is invoked and the request produced a security error for which no other information is available, then a **Security-Error** is returned with the value of the **Problem** attribute set to **no-information**. The OM attributes of the instance of **Security-Error** that is returned are as stated in R01 in 8.11. (See GA139 in 8.11.)
- 041(A) When `ds_compare` is invoked and the directory is unwilling to perform the directory operation because it was unsigned, then a **Security-Error** is returned with the value of the **Problem** attribute set to **protection-required**. The OM attributes of the instance of **Security-Error** that is returned are as stated in R01 in 8.11. (See GA140 in 8.11.)
- 042({X.500_PROTOCOLS}?A:B)When `ds_compare` is invoked and the directory operation could not be performed within the administrative constraints on the directory and no partial results are available, then a **Service-Error** is returned with the value of the **Problem** attribute set to **administrative-limit-exceeded**. The OM attributes of the instance of **Service-Error** that is returned are as stated in R01 in 8.12. (See GA141 in 8.12.)
- 043({X.500_PROTOCOLS}?A:B)When `ds_compare` is invoked and some part of the directory is temporarily too busy to perform the directory operation, then a **Service-Error** is returned with the value of the **Problem** attribute set to **busy**. The OM attributes of the instance of **Service-Error** that is returned are as stated in R01 in 8.12. (See GA142 in 8.12.)
- 044({X.500_PROTOCOLS}?A:B)When `ds_compare` is invoked and chaining is required to perform the directory operation but is prohibited by the **Chaining-Prohibited** service control, then a **Service-Error** is returned with the value of the **Problem** attribute set to **chaining-required**. The OM attributes of the instance of **Service-Error** that is returned are as stated in R01 in 8.12. (See GA143 in 8.12.)

- 045({X.500_PROTOCOLS}?A:B)When **ds_compare** is invoked and an inconsistency has been detected in the DIT that may be localized to a particular entry or set of entries, then a **Service-Error** is returned with the value of the **Problem** attribute set to **dit-error**. The OM attributes of the instance of **Service-Error** that is returned are as stated in R01 in 8.12. (See GA144 in 8.12.)
- 046({X.500_PROTOCOLS}?A:B)When **ds_compare**(, *context*,,,) is invoked and the DSA was unable to perform the request as directed (via **Operation-Progress** in the *context*), then a **Service-Error** is returned with the value of the **Problem** attribute set to **invalid-reference**. The OM attributes of the instance of **Service-Error** that is returned are as stated in R01 in 8.12. (See GA145 in 8.12.)
- 047({X.500_PROTOCOLS}?A:B)When **ds_compare**, is invoked and a DSA detected a loop within the directory then a **Service-Error** is returned with the value of the **Problem** attribute set to **loop-detected**. The OM attributes of the instance of **Service-Error** that is returned are as stated in R01 in 8.12. (See GA146 in 8.12.)
- 048({X.500_PROTOCOLS}?A:B)When **ds_compare** is invoked and the directory cannot provide a referral or partial outcome qualifier within the required scope, then a **Service-Error** is returned with the value of the **Problem** attribute set to **out-of-scope**. The OM attributes of the instance of **Service-Error** that is returned are as stated in R01 in 8.12. (See GA147 in 8.12.)
- 049({X.500_PROTOCOLS}?A:B)When **ds_compare** is invoked and the directory operation could not be performed within the time specified by the time-limit service control, and no partial results are available, then a **Service-Error** is returned with the value of the **Problem** attribute set to **time-limit-exceeded**. The OM attributes of the instance of **Service-Error** that is returned are as stated in R01 in 8.12. (See GA148 in 8.12.)
- 050({X.500_PROTOCOLS}?A:B)When **ds_compare** is invoked and a DSA without administrative authority over a particular naming context was asked to resolve a name in that context, then a **Service-Error** is returned with the value of the **Problem** attribute set to **unable-to-proceed**. The OM attributes of the instance of **Service-Error** that is returned are as stated in R01 in 8.12. (See GA149 in 8.12.)
- 051({X.500_PROTOCOLS}?A:B)When **ds_compare** is invoked and some part of the directory is not currently available, then a **Service-Error** is returned with the value of the **Problem** attribute set to **unavailable**. The OM attributes of the instance of **Service-Error** that is returned are as stated in R01 in 8.12. (See GA150 in 8.12.)
- 052(A) When **ds_compare** is invoked and one or more critical extensions were requested but not available, then a **Service-Error** is returned with the value of the **Problem** attribute set to **unavailable-critical-extension**. The OM attributes of the instance of **Service-Error** that is returned are as stated in R01 in 8.12. (See GA151 in 8.12.)
- 053({X.500_PROTOCOLS}?A:B)When **ds_compare** is invoked and some part of the directory is not willing to perform the directory operation because it requires excessive resources, then a **Service-Error** is returned with the value of the **Problem** attribute set to **unwilling-to-perform**. The OM attributes of the instance of **Service-Error** that is returned are as stated in R01 in 8.12. (See GA152 in 8.12.)
- 054({X.500_PROTOCOLS}?A:B)When **ds_compare** is invoked and some part of the directory is not willing to perform the directory operation because doing so would violate administrative policy, then a **Service-Error** is returned with the value of the **Problem** attribute set to **unwilling-to-perform**. The OM attributes of the instance of **Service-Error** that is returned are as stated in R01 in 8.12. (See GA153 in 8.12.)
- 055(B) When a system error occurs and **ds_compare** generates an instance of OM class **System-Error**, then its **Problem** attribute will identify the cause of the failure. The OM attributes of the instance of **System-Error** that is returned are as stated in R01 in 8.13. (See GA154 in 8.13.)
- 056(B) When **ds_compare**, is invoked and a transient error occurs then the affected directory operation will be retried and these errors will not be returned. (See GA155 in 8.13.)

6.5.7 Cross-References

There are no test assertions for this subclause.

6.6 Initialize

6.6.1 Synopsis

001(A) `ds_initialize` exists in the implementation. (See GA01 in 3.3.2.)

6.6.2 Description

001(A) When `ds_initialize` is invoked and completes successfully, then any necessary initialization of the interface is performed and a workspace is allocated.

002(A) When `ds_initialize` is invoked multiple times, then each workspace that is returned is distinct from other workspaces created by `ds_initialize`, but not yet deleted by `ds_shutdown`.

6.6.3 Arguments In

There are no test assertions for this subclause.

6.6.4 Arguments Out

001(A) When `ds_initialize` is invoked and completes successfully, then a handle to a workspace is returned in which OM objects can be created and manipulated.

6.6.5 Results

R01 When `ds_initialize` is invoked, then the status result that is returned is one of the following: the constant **success**, the constant **no_workspace**, the constant **invalid_workspace**, an attribute error represented by a private object of class **Attribute-Error**, a referral represented by a private object of class **Referral**, or an error represented by a private object of one of the subclasses of class **Error**. [See GA24 in 5.6.3, 001(A) in 6.6.5, and 002(A) in 6.6.5.]

001(A) When `ds_initialize` is invoked and completes successfully, then the returned value of type `ds_result_type` will be **success**. (See GA92 in 8.1.)

002(A) When `ds_initialize`, is invoked and does not complete successfully then the status result returned is **no_workspace**.

6.6.6 Errors

There are no test assertions for this subclause.

6.6.7 Cross-References

There are no test assertions for this subclause.

6.7 List

6.7.1 Synopsis

001(A) `ds_list` exists in the implementation. (See GA01 in 3.3.2.)

6.7.2 Description

001(A) When `ds_list(,name,,)` is invoked and the associated operation completes successfully, then a list of the immediate subordinates of *name* is obtained.